НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК _____

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Системне програмування

на тему: ЗАСОБИ АВТОМАТИЗАЦІЇ СКЛАДСЬКИХ ВИРОБНИЧИХ ПРОЦЕСІВ. КЕРУЮЧА СИСТЕМА

Виконав: студент ІІ курсу, групи КВ-72мп
(шифр групи)

Курилич Роман Андрійович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник ст. викладач Дробязко І.П._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент д.т.н., проф. Сімоненко В.П. _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
“Київський політехнічний інститут
імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра системного програмування і
спеціалізованих комп‘ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність 123 “Комп'ютерна інженерія” Системне програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
В.П.Тарасенко

“___” _____
2018 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ
Куриличу Роману Андрійовичу

1. Тема дисертації: Засоби автоматизації складських виробничих процесів. Керуюча частина,

науковий керівник дисертації: Дробязко Ірина Павлівна,

затверджені наказом по університету від “__” _____ 2018 року № ___.

2. Термін подання студентом дисертації: “7” грудня 2018 р.

3. Об’єкт дослідження: складські виробнич процеси

4. Предмет дослідження: засоби автоматизації складських виробничих процесів

5. Перелік задач, які потрібно вирішити: розробити засоби автоматизації, надати користувачу інтуїтивно-зрозумілий інтерфейс, забезпечити комунікацію з периферійними пристроями

6. Орієнтовний перелік ілюстративного матеріалу:

- Презентація (кількість аркушів)

7. Орієнтовний перелік публікацій:

- наукова конференція магістрантів та аспірантів «Прикладна математика та комп‘ютинг» ПМК-2018 (Київ, 14-16 листопада 2018р.);
- наукова конференція магістрантів та аспірантів «Прикладна математика та комп‘ютинг» ПМК-2018 (Київ, 14-16 листопада 2018р.);

8. Дата видачі завдання: “3” вересня 2017 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вивчення матеріалів про засоби автоматизації		
2.	Розробка засобів автоматизації		
3.	Опис існуючих рішень засобів автоматизації		
4.	Опис методів та засобів розробки		
5.	Характеристика розробленого рішення		
6.	Надання інформації про експериментальні дослідження		
7.	Підведення підсумків і результатів виконаної роботи		
8	Попередній розгляд магістерської дисертації на засіданні	26.11.2018	

.	кафедри		

Студент _____ Курилич Р. А.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Дробязко І. П.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми. Сучасний складський виробничий процес – трудомістка і витратна за часом частина виробничого циклу. Зважаючи на людський фактор, час виконання операцій на складі може нераціонально зростати. Операції відбору, сортування та інвентаризації продукції здебільшого виконуються згідно паперових документів, що надаються робітникам, і не є автоматизованими. Це негативно впливає на час виконання завдань та ймовірність помилок. Зважаючи на велику кількість компаній, які мають повний виробничий цикл, впровадження нових сучасних технологій і автоматизація складських виробничих процесів та їх моніторинг дозволить підвищити ефективність роботи працівників складу та прискорити виконання складських операцій.

Об’єктом дослідження є процеси автоматизації.

Предметом дослідження є засоби автоматизації складських виробничих процесів.

Мета роботи: підвищення ефективності засобів автоматизації складських виробничих процесів за рахунок використання сучасного програмно-апаратного забезпечення, розширення функціональності засобів автоматизації.

Іноваційність полягає системи, яка буде оброблювати статистичні показники користувачів системи та на основі отриманих даних взаємодіятиме з користувачем для пришвидшення виконання роботи, а також використовуватиме мінікомп’ютер.

Практична цінність:

1. В ході проведення аналізу досліджуваної області було визначено, що існуючі системи не надають статистичні показники виробничих процесів.

2. Було розроблено функціонал для моніторингу складських виробничих процесів та нейронну мережу для порівняльної характеристики отриманих статистичних показників.

Апробація роботи. Система для класифікація веб-сайтів на основі методів інтелектуального аналізу даних була представлена та обговорювалась на наукових конференціях магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 березня 2018 р.) та «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 14-16 листопада 2018 р.)**.**

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів і їхнє впровадження.

У першому розділі розглянуто існуючі системи автоматизації складських виробничих процесів**.**

У другому розділі розглянуто засоби розробки керуючої системи моніторингу складських виробничих процесів.

У третьому розділі описано засоби, з допомогою яких було розроблено керуючу частину засобів автоматизації та її користувацький інтерфейс.

У четвертому розділі надано інформацію про результати експериментальних досліджень системи.

У висновках представлені результати проведеної роботи.

Робота представлена на 80 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: автоматизація, складські процеси, керуюча система, інтерфейс користувача.

ABSTRACT

Actuality of theme. Modern warehouse production process - labor-consuming and time-consuming part of the production cycle. Given the human factor, the timing of operations in stock may not rationally increase. The operations of selection, sorting and inventory of products are mostly carried out in accordance with the paper documents provided to the workers, and are not automated. This negatively affects the timing of tasks and the probability of errors. Due to the large number of companies with a full production cycle, the implementation of new modern technologies and the automation of warehouse production processes and their monitoring will increase the efficiency of staffing and speed up the performance of warehouse operations.

The object of research there are automation processes .

The subject of the study There are automation tools for warehouse production processes.

The goal of the work: increasing the efficiency of automation of warehouse production processes through the use of modern software and hardware, expanding the functionality of automation tools.

Innovative is a system that will process the statistics of users of the system and on the basis of the received data will interact with the user to speed up the work , as well as use the mini computer .

Practical value:

1. During the analysis of the studied area it was determined that the existing systems do not provide statistical indicators of production processes .

2. A functional was developed for monitoring warehouse production processes and the neural network for comparative characteristics of the obtained statistical indicators.

Test work. The system for the classification of websites based on the methods of intellectual data analysis was presented and discussed at the scientific conferences of masters and postgraduates "Applied Mathematics and Computer", PMK-2018 (Kyiv, March 21-23, 2018) and "Applied Mathematics and Computing" PMK-2018 (Kyiv, November 14-16, 2018) **.**

Structure and scope of work. The master's thesis consists of an introduction, four chapters and conclusions.

The introduction gives a general description of the work, assesses the current state of the problem, substantiates the relevance of the research direction, formulates the purpose and objectives of the research, shows the scientific novelty of the results obtained and the practical value of the work, provides information on the approbation of the results and their implementation.

The first chapter examines the existing systems of automation of warehouse production processes .

In the second section the means of developing a management system for monitoring warehouse production processes are considered.

In the third section describes the means by which the control part of automation tools and its user interface were developed .

In the fourth section Information about the results of experimental research of the system was provided.

The conclusions are the results of the work.

The work is presented on 80 sheets, contains a link to the list of used literary sources.

Keywords : automation, warehouse processes, control system, user interface.

ЗМІСТ

ВСТУП

В наш час будь-яка компанія може зіткнутись з проблемою неефективного використання власних ресурсів. Це може стосуватись будь-якої ланки виробничого циклу. Однією з таких проблем є невелика кількість існуючих технічних рішень для менеджменту складського виробничого процесу.

На даний момент одним з найрозповсюдженіших способів керування робочим процесом в складських приміщеннях компанії є такий, де, здебільшого, всі операції, що мають бути виконані описуються згідно паперової документації, що видається робітнику. Очевидно, що при використанні такого способу значно збільшується час обробки отриманих вказівок та їх виконання. Більш того, існує ймовірність некоректного завершення завдання. Також слід звернути увагу на зростання споживацького інтересу суспільства, що, безперечно, веде до збільшення вимог до управління складськими приміщенням зі сторони клієнтів зокрема вимоги до оперування більшою кількістю замовлень водночас, збільшення кількості прикладних сервісів обслуговування цих замовлень на складі (пакування, комплектація тощо).

Зважаючи на вищеописані фактори, за останні два десятиліття було розроблено багато технологічних систем, котрі мають на меті спростити робочий процес працівнику шляхом надання покрокових вказівок від комп’ютерної системи автоматизації. Такі системи використовуються для багатьох складських процесів, починаючи з процесу отримання товарів на склад і завершуючи відвантаження товарів для доставки до сторони клієнта чи кінцевого споживача. Серед таких систем особливе місце займають ті, що відповідають за процес відбору товарів, який може включати етапи пакування чи комплектації, оскільки саме під час цього процесу складські приміщення зустрічаються з найбільшою проблемою –

неефективним простоєм. Такий простій зумовлений тим, що під час відбору товарів в процесі з’являється роль працівника складу, що може супроводжуватись збільшенням кількості помилок та нестабільним часом виконання поставленого.

Проте дані технології мають декілька суттєвих недоліків, наприклад таких, як: висока вартість імплементації в складське приміщення, відсутність можливості моніторингу ефективності роботи працівників тощо.

З огляду на вищезазначене, актуальним є створення засобів автоматизації складських виробничих процесів, керуюча система буде здійснювати моніторинг робочого процесу на складі, а також буде комунікувати за визначеним протоколом з виконавчою системою, надаючи потрібні вказівки.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАДАЧІ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

1.1. Загальна характеристика складських виробничих процесів

Сьогодні складські операції є обов’язковим етапом на шляху будь-якого товару від виробника до кінцевого користувача. Тож склади повинні бути забезпечені відповідними технічними засобами та ефективно їх застосовувати при управлінні як внутрішніми, так і зовнішніми процесами.

Складування включає ряд заходів, пов'язаних з тимчасовим оформленням, зберіганням, збиранням, транспортуванням, обслуговуванням, контролем та відвантаженням матеріального інвентарю.

Основний процес складування можна поділити на чотири основні етапи:

1) Отримання та розташування;
2) Зберігання;
3) Збирання та пакування;
4) Відвантаження товарів, що зберігаються.

Основні завдання, що виконуються в рамках першого та другого етапів, включають розвантаження, ідентифікацію, сортування, перевірку якості та кількості, підготовку товарів для зберігання та їх переміщення на зберігання.

Третій етап полягає у збиранні (відборі) та пакуванні товарів у визначеній кількості з визначеними характеристиками перед відвантаженням відповідно до замовлення.

Останній етап процесу – відвантаження замовлення зі складу. На цьому етапі елементи упаковуються та формуються у транспортні одиниці, перевіряються та завантажуються в транспортний засіб.

У порівнянні з іншими етапами, збирання, відбір та пакування товарів використовує найбільше складських ресурсів і може задіяти близько 60% або більше складських працівників [1].

Збір замовлень здійснюється в окремому місці, але також може відбуватися в зоні зберігання. Основний спосіб збирання передбачає збирання відповідно до замовлень. Під час збирання всі елементи із замовленого асортименту відбираються з комірок їх зберігання. Інший спосіб – збирання відповідно до асортименту – вимагає проходження етапу збирання з вилученням предметів для декількох замовлень. На другому кроці відібрані елементи розподіляються на індивідуальні замовлення. Впродовж цього етапу працівник повинен бути забезпечений інформацією про індекс товару, назву, необхідну кількість та детальне розташування.

Для полегшення роботи працівнику складу під час збирання товарів, на складах використовується спеціальний спосіб метод розташування товарів ще на етапі зберігання. Такий спосіб метод передбачає розташування товарів таким чином, щоб ті, що замовляються частіше (група товарів А), були розміщені якомога ближче, у випадку багаторівневих стелажів – нижче до зони пакування та відвантаження. Групи В і С відповідно до частоти замовлень розташовуються далі. Таким чином, робітник, який збирає товар, не повинен здійснювати зайві переміщення по складу.

Маючи такі комплексні етапи в процесі використання складського приміщення, для коректного та ефективного функціонування потрібно застосовувати належні технології та ефективне управління складськими процесами. Розвиток обох цих елементів має еволюційну форму. Управління складськими процесами звичайно розвивається з часом та зумовлене побудовою чи перебудовою логіки процесів на складі, в той

час як розвиток технологій зумовлений збільшенням кількості замовлень та потребою зменшення періоду перебування товару на складі.

Склади малого та середнього бізнесу, в переважній більшості, використовують паперовий спосіб при збиранні та пакуванні товару. Це не вимагає додаткових витрат на спеціальне програмне забезпечення та технічне обладнання. Та зі зростанням потоку товарів, збільшенням різновидів та кількості самих товарів, яке спостерігається в теперішній час, оскільки суспільство почало більше споживати, менеджмент складських приміщень починає розмірковувати над імплементацією нових технологій, які дадуть змогу збільшити їх ефективність, не збільшуючи кількість працівників на складі.

Складські приміщення в сьогоденні все частіше почали відмовлятись від традиційного паперового відбору замовлень, оскільки він має ряд особливостей, які сповільнюють роботу складу і знижують її ефективність. Так, оператор, крім власне самого відбору, повинен виконувати багато додаткових дій. Наприклад, час витрачається на перевірку документів, виправлення можливих помилок, пошук місцезнаходження необхідної комірки у сховищі, зайве пересування між стелажами і комірками. При паперовому відборі трапляються такі помилки, як наприклад неправильний підрахунок елементів, невірно вибрана збирачем кількість або відібраний інший елемент. Підраховано, що на ці дії збирачі витрачають приблизно 80% свого робочого часу [1].

1.2. Аналіз основних підходів до автоматизації складських виробничих процесів на етапі відбору та комплектації замовлення

Pick by voice

Система, що працює за принципом Pick by voice, в своїй основі використовує технології розпізнавання голосу для складських операцій комплектації, інвентаризації та інших. Зараз багато компаній розвивають

подібні системи, щоб їх використовувати під час виконання всіх складських процесів з метою уникнення помилок, спричинених людським фактором.

Система на основі Pick by voice забезпечує голосовий інтерфейс, що дозволяє користувачам спілкуватися з хост-системами через головну гарнітуру і портативні термінали. Система надає голосові інструкції працівникам під час виконання повсякденних завдань, забезпечуючи при цьому можливість відслідковування всього процесу.

Комплектувальник отримує через навушники комп'ютеризовані мовні інструкції і дає відповідні команди через мікрофон (рис. 1.1).

Рисунок 1.1 – Загальна схема роботи системи на основі Pick by voice

Функціонування системи можна описати наступною послідовністю кроків:

1. Наряди на збірку замовлень, тобто комплектувальні листи, завантажуються на портативний термінал і після обробки передаються через навушники в мовній формі. Після кожної дії комплектувальника інформація про хід виконання завдання надсилається хост-системам.
2. Система управління складськими процесами надсилає вказівки в голосову гарнітуру.
3. Гарнітура перетворює отримані вказівки в голосові повідомлення.

4. Комплектувальник отримує голосові команди за допомогою навушників.
5. Послідовно виконуючи команди, оператор підтверджує виконання голосовою командою і отримує голосові вказівки про наступний відбір.

На сьогоднішній день на ринку є чимало компаній, які займаються розробкою таких систем. Переважна більшість із них зазначає про підвищення наступних показників роботи складу при використанні Pick by voice [1, 2]:

1. Продуктивність – на 10 - 35%;
2. Точність комплектації – до 99.99%

Також серед переваг є прозорість ходу виконання роботи в режимі реального часу (за рахунок відстеження за допомогою спеціального обладнання), простий інструктаж для нових недосвідчених працівників, руки й очі оператора звільняються, що забезпечує меншу травмованість персоналу.

До головного недоліку системи, котра працює за принципом Pick by voice, можна віднести значну вартість складських технічних засобів, їх імплементації та підтримки, а також технічного обладнання безпосередньо для працівників.

Система з використанням RF-Scanning

Системи, що використовують RF-Scanning, є одними з найбільш розповсюджених на складах систем автоматизації. RF означає radio-frequency (з англ. радіочастота). Тобто під використанням RF сканування мається на увазі використання працівниками складу радіочастотних сканерів, які надають їм змогу зчитувати штрих-коди з етикеток товару.

Збірка зі сканером – це процес ручної збірки замовлень з використанням радіо-сканерів. Оператору на екрані сканера надається

інформація, які саме товари або продукти треба зібрати. Одна така система може бути використана в декількох зонах та процесах складу. Під час складання товарів сканується штрих-код товару і його місце розташування для підтвердження інформації. Це забезпечує точність збору і продуктивність. Іншою важливою і унікальною особливістю системи є гнучкість. Використання системи збирання замовлень з радіо-сканерами дозволяє легко використовувати її в окремих зонах збірки, незалежно від числа місць розташувань товарів.

Система на основі Pick to light

Pick to light являє собою принцип роботи цифрової системи відбору товарів, який виключає використання паперових документів і дозволяє контролювати всі пересування товарів електронним способом. Інформація про замовлення в межах компанії передається в електронному вигляді у WMS-систему. Збирач отримує конкретні вказівки щодо виконуваних операцій за допомогою світлових модулів, розміщених біля кожної комірки зберігання товару. Світлові модулі вказують на місце та кількість товару, який необхідно відібрати (рис. 1.2).

Система дозволяє проводити операції відбору, сортування та інвентаризації товарів. Її можна використовувати в дистрибуції (в складах і дистрибутивних центрах будь-якої спрямованості) і на виробництві (на виробничих складах і лініях збірки).

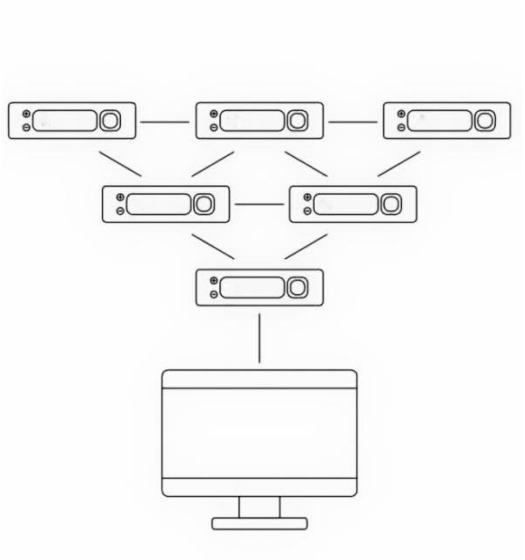

Рисунок 1.2 – Система Pick to light

Звичайно система типу Pick to light є першим варіантом для переходу від паперового відбору до інших способів або ж є подальшим розширенням технологічних можливостей складу після використання методів RF-Scanning.

У порівнянні з паперовим способом відбору, система Pick to light надає наступні переваги:

3. Підвищення продуктивності: в середньому продуктивність підвищується на 50% і більше, а в деяких ситуаціях – до 200% [1];
4. Скорочення кількості помилок відбору в середньому на 70 - 90% [2];
5. Висока точність заданих операцій;
6. Простота, надійність і гнучкість операцій;
7. Зменшення робочого навантаження і вимог при навчанні працівників;
8. Зменшення кількості працівників;
9. Налаштування системи з урахуванням індивідуальних умов роботи;
10. Відстеження стану замовлення в реальному часі;

11. Відстеження змісту кожної комірки в реальному часі;
12. Короткий період окупності системи.

Саме такі переваги системи з використанням Pick to light зумовили розглянути її існуючі рішення – від компаній Lightning Pick [3], PicktoLight Systems [4], Bastian Solutions [5] та інших.

Ці рішення мають один суттєвий недолік, як і у випадку використання Pick by voice, – перешкодою для впровадження таких засобів автоматизації для складських приміщень малого та середнього бізнесу є значна вартість технічних засобів та обладнання, а також їх імплементації. Імплементація в цілому потребує чимало часу, оскільки налаштування програмних засобів на виконання конкретних процесів того чи іншого складу передбачає також навчання менеджера, котрий надалі буде керувати програмою. А також, серед розглянутих варіантів немає такого, який пропонує застосування моніторингу робочого процесу.

1.3 Порівняльний аналіз існуючих рішень

Сьогодні день існує багато рішень для засобів автоматизації складських виробничих процесів. Нижче будуть наведені найпопулярніші з них.

ABC Inventory Software

ABC Inventory Software – це безкоштовне програмне забезпечення для інвентаризації для малого та середнього бізнесу, розроблене Almyta Control System. Існує його розширена функціональна версія, проте для користування нею потрібно вносити щомісячну передплату [5].

Серед функцій, які надає ABC Inventory Software, є наступні:

- Базові: створення профілю користувача, розмежування користувачів за правами доступу, налаштування графічного

інтерфейсу (кольорова гамма), експорт даних в формати Excel та HTML та інші;

- Виробничі: додавання збірки за замовленням, додавання збірки для комплектування, використання декількох робочих станцій, підключення системи типу Pick to light.

Серед переваг, які зазначено на сайті продукту [5], вказано такі: немає обмежень у кількості записів про товар в базу даних, базовий функціонал абсолютно безкоштовний, необмежена кількість користувачів, багатомовність тощо.

До недоліків можна віднести відсутність стеження за процесом виконання збірки замовлення; продукт працює тільки з операційними системами серії Windows; немає можливості моніторингу та здійснення порівняльного аналізу часу, за який працівник виконав створення збірки чи відбір товару для замовлення. А також система не надає можливості додавати відображення вигляду (фотографії) товару чи окремого елементу збірки.

Zezoo, ZPanel

Адміністративна панель Zezoo, ZPanel – це веб-орієнтоване програмне забезпечення для складання та виконання замовлень для електронної комерції, яке використовується для: управління та організації замовлень та замовників; відстеження інвентарю; інтеграції з ринками, такими як Amazon; управління поверненнями; управління складом та інвентарем; надання клієнтам бізнес-звітів; інтегрування з бухгалтерським програмним забезпеченням, таким як Quickbooks.

ZPanel надає такі можливості:

- Повне стеження й контроль замовлень;
- Створення та редагування збірки;
- Інтеграція з такими інтернет маркетами, як Amazon;

- Відстеження перевезень, інтеграція з FedEx, UPS;
- Багатомовність системи;
- Можливість під’єднання декількох робочих станцій;
- Підключення системи типу Pick to light та Pick by voice тощо.

До переваг Zpanel можна віднести її багатофункціональність, можливість інтеграції з інтернет маркетами (Amazon та інші) та службами доставки (FedEx, UPS). Також слід зазначити, що Zpanel доступна для найпоширеніших операційних систем Windows, MacOS, Linux тощо. Система підходить для великого бізнесу, оскільки має цілодобову підтримку користувачів та велику кількість додаткових сервісів.

Серед недоліків таких засобів автоматизації є їх значна вартість та необхідність щомісячного внеску; загалом вони орієнтовані на великий бізнес. Також Zpanel не надає можливості моніторингу робочого процесу та часу, який працівник використовує на виконання тієї чи іншої збірки.

1.3. Обґрунтування теми магістерської дисертації та постановка задачі

За останні роки в цілому світі спостерігається тенденція зростання попиту на покупки найрізноманітніших товарів, більшість з яких на своєму довгому шляху до кінцевого споживача проходять через складські приміщення та процеси. В свою чергу, власники складів повинні надавати сервіс належної якості для зберігання, розміщення, а в багатьох випадках і для комплектації товарів. Оскільки виконання всіх процесів вимагає швидкості й безпомилковості, чого, звичайно, працівнику важко досягти без використання спеціальних технологій, було вирішено розробити ефективні засоби автоматизації складських виробничих процесів.

Головними відмінностями таких засобів автоматизації від аналогів мають стати: максимально спрощений інтерфейс задля інтуїтивного розуміння (сприйняття) й зручного користування засобами навіть

недосвідченими користувачами ПК, а також можливість моніторингу якості та швидкості роботи працівників складу.

Головною завданням магістерської дисертації є розробити керуючу систему автоматизації складських виробничих процесів. Для цього потрібно вирішити наступні задачі:

1. Визначити загальну архітектуру системи та функціональність її складових.
2. Здійснити оптимальний вибір технічних та апаратних засобів для її реалізації.
3. Реалізувати систему, що забезпечує наступні функціональні можливості:
   1) Створення наборів комплектацій товару з можливістю їх редагування;
   2) Моніторинг ефективності роботи працівників;
   3) Візуалізація елементів і результатів комплектації;
   4) Забезпечити різні режими роботи системи;
   5) Забезпечити зручні засоби комунікації між складовими системи та з користувачем.

1.4 Висновки до розділу 1

Проведено аналіз існуючих рішень систем автоматизації складських виробничих процесів, визначено їх особливості.

Обгрунтовано необхідність побудови власної системи автоматизації, що враховуватиме недоліки відомих систем, працюватиме за принципом Pick to light, матиме додаткові функціональні можливості і менші витрати на програмне й апаратне забезпечення.

Сформульовано основні задачі, які потрібно вирішити для побудови засобів автоматизації складських виробничих процесів.

2. СПОСОБИ ТА ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ ЗАСОБІВ АВТОМАТИЗАЦІЇ

2.1 Основні компоненти засобів автоматизації складських виробничих процесів

На основі аналізу складських виробничих процесів та керуючих засобів їх автоматизації перед розробкою було вирішено виділити в засобах автоматизації дві взаємодіючі частини: керуючу й виконавчу системи (рис. 2.1). Керуюча система повинна визначати логіку операцій, надавати вказівки виконавчій системі, відповідати за взаємодію з користувачем, в той час, як виконавча система повинна реалізовувати визначену керуючою системою логіку роботи за допомогою апаратних пристроїв.

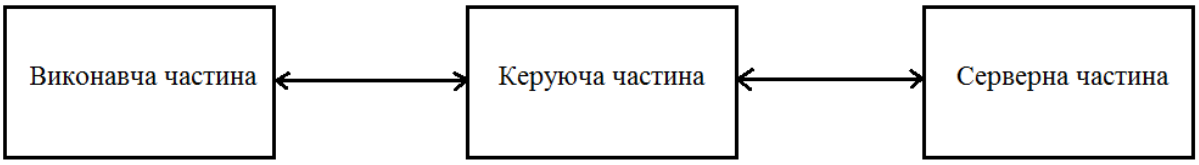

Рисунок 2.1 – Загальна архітектура засобів автоматизації складських виробничих процесів

В цій роботі розглядається побудова керуючої системи, яка матиме такі складові:

- клієнтська частина, яка визначає логіку прийняття рішень та їх застосування під час виконання різних операцій, а також взаємодіє з користувачем через графічний інтерфейс;
- серверна частина з системою управління базами даних та іншими джерелами даних тощо.

Керуюча система повинна виконувати наступні функції:

1. Проводити моніторинг ефективності роботи працівників;
2. Призупиняти роботу на певний час за бажання користувача;

3. Комунікувати з виконавчою системою;
4. Записувати необхідну інформацію й зберігати її на сервері.

2.2 Застосування мінікомп’ютера та операційної системи

Однією з основних вимог до розробки засобів автоматизації складських виробничих процесів було зменшити енергозатратність системи та її собівартість. Для виконання такої вимоги потрібно зменшити витрати на технічне обладнання та його обслуговування. У зв’зяку з цим, вирішено портувати керуючу систему автоматизації складських виробничих процесів на мінікомп’ютер [4], оскільки вартість апаратного й програмного забезпечення мінікомп'ютера значно менша порівняно з персональним комп’ютером.

Завданням є оптимізувати налаштування мінікомп’ютера для керуючої системи засобів автоматизації складських виробничих процесів з мінімізацією, в той же час, необхідних технічних ресурсів, що сприятиме зниженню загальної собівартості системи.

Мінікомп’ютер Orange Pi

На сьогоднішній день на ринку представлено чимало мінікомп’ютерів, серед яких є моделі різної цінової категорії та різних можливостей. Найвідомішими є мінікомп’ютери серії Raspberry Pi, Cubieboard, Orange Pi тощо. Останній з них виділяється меншою вартістю при приблизно тих самих характеристиках та потужності [5]. Orange Pi One побудований на базі однієї з найпопулярніших платформ – Allwinner H3 (рис. 2.2), що з'явилася в 2014 році.

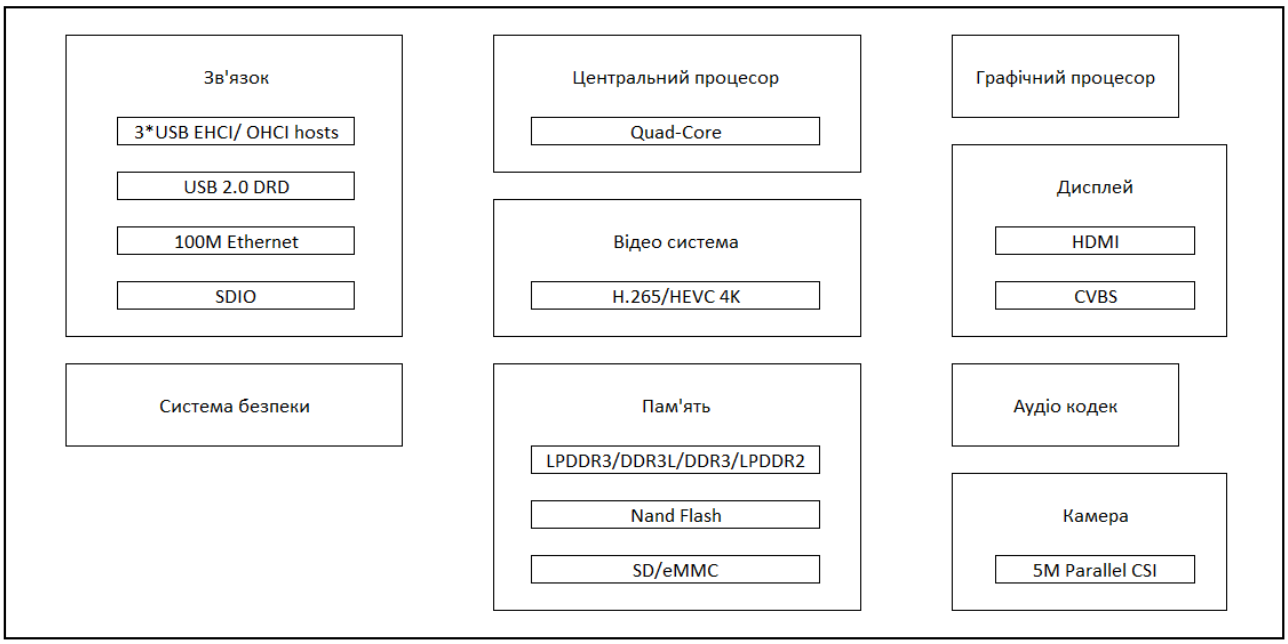

Рисунок 2.2 – Блок-схема платформи AllWinner H3

До її складу входять чотири ядра Cortex-A7 (1,2 ГГц) і відеоядро Mali 400MP2 (600 МГц). Контролер оперативної пам’яті підтримує чіпи стандартів DDR2 і DDR3 зі звичайною і зниженою напругою живлення. Мінікомп'ютер має 512 МБ DDR3 оперативної пам’яті (2 чіпи по 256 МБ кожен).

Карта пам'яті microSD (XC) з максимальним об’ємом 64 ГБ підключається через слот з інтерфейсом SDIO. Користувачеві доступні мережевий порт Ethernet (100 Мбіт / с) або модуль Wi-Fi (залежно від обраної комплектації) і концентратор USB 2.0. Orange Pi One має один повнорозмірний порт USB 2.0 Af і один mini-USB з підтримкою OTG. Споживча потужність досягає 10 Вт. Рекомендується використовувати мережевий адаптер з вихідною напругою 5 В і силою струму 2 А. Зображення можна виводити на порт HDMI, а отримувати по інтерфейсу CSI з камери з дозволом до 5 Мп. Orange Pi One має 40 програмованих контактів загального призначення (GPIO).

Як і всі одноплатні комп’ютери з процесорами архітектури ARM, Orange Pi One підтримує широкий набір операційних систем. Розміри Orange Pi One складають 69 × 48 мм, а маса – всього 36 г.

Операційна система

Засоби автоматизації складських виробничих процесів, які розробляються, будуть використовувати мінікомп’ютер Orange Pi, а не стандартні персональні комп’ютери. Для такого рішення існує чимало операційних систем, які можна встановити на мінікомп’ютер, серед яких є Android, Debian Desktop, Ubuntu Desktop, Arch Linux, Raspbian, Armbian тощо. Для використання в даній магістерській дисертації було обрано Armbian через простоту її встановлення та базу, на якій вона побудована – Debian.

Armbian – це комп'ютерна операційна система на базі Debian та Ubuntu для плат розробок ARM.

Особливості Armbian:

- Проста. BASH оболонка, стандартні утиліти Debian / Ubuntu. Загальні та специфічні функції можуть бути з мінімалістичною корисною системою меню. Логін можливий за допомогою серійного, HDMI / VGA або SSH.
- Легка. Немає вірусів або шпигунських програм. Спеціальні утиліти абсолютно опціональні. Підходить для новачків та професіоналів.
- Оптимізована. Розповсюджений образ стискується до реального розміру даних і складає приблизно 1 ГБ. Розмір оптимізований для використання SD-карти.
- Швидка. Плати оптимізовані на рівні ядра та користувачів. Оптимізація DVFS, кешування журналу пам'яті, кешування

пам'яті профілю браузера, настройка використання підкачки, затримка сміття. Наша система працює майже лише для читання, і це є одним із самих швидких Linux для багатьох платформ розробки практично у кожному випадку.

- Безпечна. Рівень безпеки знаходиться на рівні безпеки Debian і може бути покращеним за допомогою конфігураційної утиліти. Це забезпечує гарну відправну точку для комерційного або домашнього використання. Кожна офіційна стабільна конструкція ретельно протестована. Зображення є прямим базою для всіх 3-сторонніх будівельників.
- Підтримується. Надання довгострокових оновлень, виправлень безпеки, документації, підтримки користувачів.
- Розумна. Глибоке розуміння того, як працюють плати, як працює операційна система, а також те, як апаратні засоби повинні бути розроблені, щоб працювати краще. Розробники Armbian брали участь у дизайні плати. Досвід роботи в Linux з початку 90-х. Спеціалізується на розробках ARM з 2013 року.
- Відкрита. Opensource скрипт збірки та розробки ядра, обслуговування та розповсюдження для більш ніж 30 різних ARM і ARM64 ядер Linux. Потужні інструменти для створення та розробки програмного забезпечення. Може працювати в повністю паралельному режимі. Може запускатись під Docker.

2.3 Загальні архітектурні рішення

Клієнт-серверна архітектура

Архітектура клієнт-сервер є архітектура комп'ютерної мережі, в якій багато клієнтів (віддалені процесори) запитують та отримують сервіс із централізованого сервера (хост-комп'ютер). Клієнтські комп'ютери

забезпечують інтерфейс, який дозволяє користувачеві комп'ютера запитувати сервери та відображати результати, які сервер повертає (рис. 2.2). Сервери чекають надходження запитів від клієнтів, а потім відповідають на них.

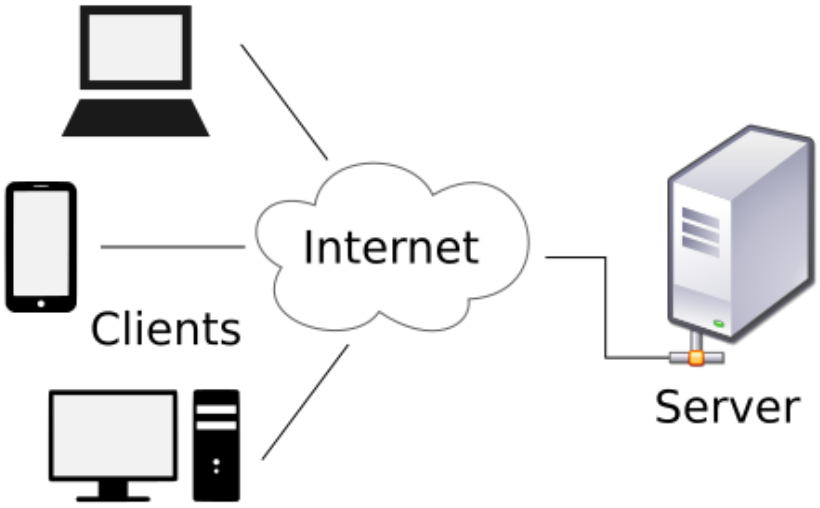

Рисунок 2.2 – Клієнт-серверна архітектура

Найчастіше сервер надає стандартний прозорий інтерфейс для клієнтів, щоб клієнти не потребували знання про специфіку системи (наприклад, апаратного та програмного забезпечення), яка надає послугу. Клієнти часто розташовуються на робочих станціях або на персональних комп'ютерах, а сервери знаходяться в інших місцях мережі, як правило, на більш потужних машинах. Ця обчислювальна модель особливо ефективна, коли клієнти та сервер мають окремі завдання, які вони виконують за замовчуванням.

Оскільки як клієнтські, так і серверні комп'ютери розглядаються як інтелектуальні пристрої, модель клієнт-сервер повністю відрізняється від старої моделі "мейнфрейма", в якій централізований комп'ютер основного комп'ютера виконував всі завдання для пов'язаних з ним "німих" терміналів.

Основними складовими цієї архітектури є клієнт та сервер. Клієнтом називається однокористувацька робоча станція, яка надає послуги презентації, послуги баз даних та зв’язку з ними, а також інтерфейс для взаємодії з користувачем для отримання бізнес-потреб. Сервером є один або декілька багатокористувацьких процесорів із більшою кількістю спільної пам'яті, яка забезпечує підключення та послуги баз даних, а також інтерфейси, що стосуються бізнес-процедур.

Основними перевагами використання даної архітектури є:

- Покращений обмін даними. Дані зберігаються звичайними бізнес-процесами, а маніпулювання на сервері доступні для призначених користувачів (клієнтів) за авторизованим доступом. Використання мови Structured Query Language (SQL) підтримує відкритий доступ з усіх аспектів клієнта, а також прозорість в мережевих службах, що показують, що подібні дані діляться серед користувачів.
- Інтеграція послуг. Кожен клієнт отримує доступ до корпоративної інформації через інтерфейс ПК, усуваючи необхідність входити в режим термінала або інший процесор. Настільні інструменти, такі як електронна таблиця, презентації Power Point тощо, можуть використовуватися для обробки корпоративних даних за допомогою баз даних та серверів додатків, резидентів мережі, для отримання значущої інформації.
- Спільні ресурси серед різних платформ. Програми, що використовуються для клієнт-серверної моделі, будуються незалежно від апаратної платформи, що забезпечує відкрите обчислювальне середовище, дозволяючи користувачам отримувати послуги клієнтів і серверів (база даних, додатки, зв'язку сервери)
- Взаємодія даних. Всі інструменти розробки, що використовуються для клієнтських / серверних програм, мають доступ до сервера баз даних за допомогою SQL, стандартного стандарту визначення

даних та мови доступу, що є корисним для послідовного управління корпоративними даними. Розширені продукти бази даних дозволяють користувачеві / додатку отримувати об'єднаний вигляд корпоративних даних, розпорошених за кількома платформами. Замість однієї цільової платформи це забезпечує цілісність бази даних з можливістю виконання оновлень у кількох місцях, що забезпечують рециклінг якості та відновлення.

- Безпека. Сервери краще контролюють доступ та ресурси, щоб забезпечити доступ або їхню зміну лише уповноваженими клієнтами.

Шаблон програмування

Model View Controller (MVC) – це архітектурний стиль, який зазвичай використовується у при розробці будь-якого виду додатків. Він забезпечує три основні шари; модель, перегляд і контролер. Багато розробників використовують MVC як стандартний шаблон дизайну. Це повноцінна структура.

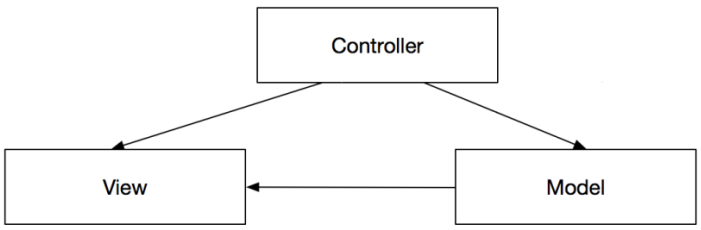

Рисунок 2.4 – Шаблон проектування MVC

MVC забезпечує три типи класів:

- Модель – класи моделі використовуються для реалізації логіки областей даних. Ці класи використовуються для отримання, вставки або оновлення даних у базу даних, пов'язаної з нашою програмою.

- Вигляд – класи вигляди використовуються для підготовки інтерфейсу програмного продукту. За допомогою цього інтерфейсу користувачі взаємодіють з нашою програмою.
- Класичні контролери – контролери використовуються для відповіді на запити користувача. Класи контролерів виконують запити користувачів на дії. Ці класи працюють з класами моделі і вибирають відповідний вигляд, який повинен відображатися користувачеві відповідно до запитів користувача.

Архітектура шаблону MVC в основному складається з трьох шаруватих архітектур. Він відокремлює характеристики застосування. Його перший шар пов'язаний з логікою введення користувача, другий рівень пов'язаний з бізнес-логікою, а третій - для реалізації логіки користувача інтерфейсу. MVC забезпечує дуже вільну зв'язок між цими трьома рівнями. Шаблон MVC використовується для визначення розташування кожної логіки у додатку. Шаблони MVC забезпечують паралельний розвиток. Це означає, що кожен шар програми незалежно один від одного, тобто три розробника можуть працювати на одній програмі одночасно [3]. Один розробник буде працювати над логікою введення користувача (логіка контролера), інший розробник буде працювати над логікою користувацького інтерфейсу (перегляду), а третій розробник буде працювати над бізнес-логікою (моделлю) одночасно.

Переваги архітектури MVC

- Архітектура MVC допомагає нам контролювати складність застосування, розділяючи його на три компоненти, тобто модель, вид і контролер.
- MVC не використовує серверні форми, тому ідеально підходить для тих розробників, які хочуть повністю контролювати поведінку своїх програм.

- Тестовий підхід до розробки підтримується архітектурою MVC.
- MVC використовувати фронт контролера шаблон. Шаблон переднього контролера обробляє кілька вхідних запитів за допомогою одного інтерфейсу (контролера). Фронтовий контролер забезпечує централізований контроль. Нам потрібно налаштувати лише один контролер у веб-сервері замість багатьох.
- Фронт-контролер забезпечує підтримку посиленої комунікації маршрутизації для розробки нашого веб-додатка.

Коли відбувається розділення логіки застосування продукту на три завдання (логіка вхідних даних, бізнес-логіка, логіка інтерфейсу), тестування цих компонентів стане дуже простим. Тестування такого продукту відбуватиметься дуже швидко і гнучко, тому що можна використовувати будь-яку структуру модульного тестування, сумісну з системою MVC. Це розширювана і підключена система. Конструювання компонентів програми може відбуватись таким чином, щоб вони були легко замінні або легко модифіковані.

2.4 Технології та мови програмування

Перш за все при створення будь-якого продукту потрібно обрати мову програмування для розробки. Звичайно мову програмування обирають відповідно до мети розробки та вимог до програмного продукту і планів подальшого розвитку. Для розробленої системи вибір проводився серед універсальних мов програмування, які надаватимуть потрібний інструментарій, бібліотеки та матимуть певні фреймворки, щоб забезпечити найбільш точне виконання поставлених задач.

Java

Java — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм.

Python

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [7].

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- кросплатформеність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і написане мовою Python;
- зручний для розв'язання математичних задач (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

На відміну від мови програмування Java, Python надає зручні утиліти для керування віртуальними портами, які забезпечують зв’язок з виконавчою системою. Це є вагомим чинником при виборі мови програмування для реалізації керуючої системи.

Крім того, розмір Java Virtual Machine (JVM), значно перевищує розмір інтерпретатора Python, а в умовах роботи на мінікомп’ютері цей фактор відіграє значну роль. Саме тому для розробки керуючої системи обрано мову програмування Python.

2.5 Технології та засоби розробки графічних інтерфейсів

Оскільки керуюча частина буде використовуватись кінцевими користувачами – працівниками на складі, потрібно обрати засоби розробки графічного інтерфейсу, котрі надали би можливість створити інтуїтивно-зрозумілий та простий в користуванні інтерфейс.

Проаналізуємо відомі засоби та підходи до створення графічних інтерфейсів користувача.

Інтерфейс користувача

Дизайн інтерфейсу користувача - це процес створення користувацького інтерфейсу у програмному забезпеченні або на комп'ютеризованих пристроях з основним акцентом на зовнішній вигляд або стиль. Дизайнери прагнуть створити такий вигляд, який користувачеві буде простим та приємним у використанні. Користувацький інтерфейс є першим етапом ознайомлення користувача з продуктом, саме цьому його дизайн є важливим завданням.

Дуже важливо визначити необхідні головні характеристики хорошого інтерфейсу користувача. Головні серед яких є:

- Швидкість навчання. Хороший інтерфейс користувача повинен бути легко вивченим. Швидкість навчання ускладнюється складним синтаксисом та семантикою процедур командного випуску. Хороший інтерфейс користувача не повинен вимагати, щоб його користувачі запам'ятовували команди. Також не слід запросити користувача пам'ятати інформацію з одного екрана на інший під час виконання різних завдань за допомогою інтерфейсу.
- Швидкість використання. Швидкість використання користувальницького інтерфейсу визначається часом і зусиллями користувача, необхідними для ініціювання та виконання різних команд. Ця характеристика інтерфейсу кілька разів називається

підтримкою продуктивності інтерфейсу. Це вказує, як швидко користувачі можуть виконувати свої передбачувані завдання. Час, необхідний для запуску та виконання різних команд, повинен бути мінімальним. Це можна досягти шляхом ретельного оформлення інтерфейсу. Наприклад, інтерфейс, який вимагає, щоб користувачі вводили тривалі команди або включали в себе рух миші на різні області екрану, які є різними для випуску команд, можуть сповільнити швидкість роботи користувачів. Найчастіше використовувані команди повинні мати найменшу довжину або бути доступними у верхній частині меню, щоб мінімізувати рухи миші, необхідні для видачі команд.

- Швидкість відкликання. Як тільки користувачі навчаться використовувати інтерфейс, слід максимально збільшити швидкість, з якою вони можуть нагадати процедуру випуску команд. Ця характеристика дуже важлива для користувачів з нестабільністю. Швидкість відкликання покращується, якщо інтерфейс базується на деяких метафорах, символьних процедурах видачі команд та інтуїтивно зрозумілих іменах команд.
- Зворотній зв'язок Хороший користувацький інтерфейс повинен забезпечити зворотний зв'язок для різних дій користувача. Особливо, якщо будь-який користувацький запит вимагає обробки більше, ніж кілька секунд, користувач повинен бути поінформований про стан обробки його запиту. Якщо тривалий час відсутня будь-яка відповідь з комп'ютера, користувач-початківець може навіть почати паніку в процесі відновлення / завершення роботи. У разі потреби користувач повинен періодично інформувати про прогрес, досягнутий при обробці його команди.

- Попередження помилок. Хороший інтерфейс користувача повинен мінімізувати обсяг здійснення помилок під час ініціювання різних команд. Коефіцієнт помилки інтерфейсу може бути легко визначений шляхом контролю за помилками середніх користувачів при використанні інтерфейсу.

PyQT

PyQt – набір інструментів графічного фреймворка Qt для програмування мовою Python, виконаний у вигляді розширення для цієї мови.

PyQt розроблений британською компанією Riverbank Computing. PyQt працює на всіх платформах, які підтримує Qt: Linux та інших UNIX-подібних ОС, Mac OS X і Windows. Інструменти PyQt поширюються за ліцензіями GPL (2 і 3 версії) і комерційної.

PyQt практично повністю реалізує всі можливості Qt. А це більше 600 класів, більше 6000 функцій і методів, включно з:

- набір віджетів графічного інтерфейсу;
- стилі віджетів;
- доступ до бази даних за допомогою SQL (ODBC, MySQL, PostgreSQL, Oracle);
- QScintilla, заснований на Scintilla - віджет текстового редактора;
- підтримка інтернаціоналізації (i18n);
- парсер XML;
- підтримка SVG;
- інтеграція з WebKit, рендерінг HTML;
- підтримка відтворення відео та аудіо потоків тощо.

PyQt також включає Qt Designer (Qt Creator) - дизайнер графічного інтерфейсу користувача. Програма pyuic генерує Python код з файлів,

створених у Qt Designer. Це робить PyQt дуже корисним інструментом для швидкого прототипування. Крім того, можна додавати нові графічні елементи керування, написані на Python, в Qt Designer. Qt Designer є кросплатформним засобом компанування макетів та форм графічного інтерфейсу користувача. Він дозволяє швидко проектувати віджети та діалоги, застосовуючи екранні форми з поєднанням тих же віджетів, які будуть використовуватися в додатку. Форми, створені з Qt Designer, є повністю функціональними, а також можуть бути переглянуті в режимі реального часу.

Tkinter

Tkinter – це бібліотека для розробки графічних інтерфейсів мовою Python з набором інструментів Tk GUI. Це стандартний інтерфейс Python для Tk GUI toolkit, і є стандартним графічним інтерфейсом мови програмування Python. Tkinter входить до складу стандартних наборів Linux, Microsoft Windows і Mac OS X в інсталяціях Python.

Як і у більшості інших сучасних Tk відгалужень, Tkinter реалізується як обгортка Python навколо повного інтерпретатора Tcl, вбудованого в інтерпретатор Python. Код Tkinter перекладений в команди Tcl, які надсилаються цьому вбудованому інтерпретатору, що дозволяє змішувати Python і Tcl в одній програмі.

Бібліотека призначена для організації діалогів в програмі за допомогою віконного графічного інтерфейсу (GUI). У складі бібліотеки присутні загальні графічні компоненти, такі ж як і в PyQT:

- Toplevel. Вікно верхнього рівня (кореневий віджет) ;
- Tk;
- Frame. Рамка, містить в собі інші візуальні компоненти;
- Label. Лейбл, показує деякий текст або графічне зображення;
- Entry. Поле введення тексту;

- Canvas. Полотно, може використовуватися для виведення графічних примітивів, наприклад, для побудови графіків;
- Button. Кнопка, проста кнопка для виконання команди і інших дій;
- Radiobutton. Перемикач, один із альтернативних значень деякої змінної. Звичайно діє в групі. Коли користувач вибирає будь-які опції, з раніше обраного в цій же групі елемента вибір знімається;
- Checkbutton. Прапорець – кнопка, що має два стани, при натисканні змінює стан на протилежне;
- Scale. Шкала, дозволяє задати числове значення шляхом переміщення движка;
- Listbox. Список, показує список, з якого користувач може виділити один або кілька елементів;
- Scrollbar. Полоса прокрутки, може використовуватися разом з деякими іншими компонентами для їх прокрутки;
- OptionMenu;
- Spinbox;
- LabelFrame;
- PanedWindow;
- Menu. Меню, служить для організації спливаючих (popup) і спадаючих (pulldown) меню;
- Menubutton. Кнопка-меню, використовується для організації pulldown-меню;
- Message. Повідомлення, дозволяє загортати довгі рядки і легко змінює свій розмір;

- Text. Форматований текст, дозволяє показувати, редагувати і форматувати текст з використанням різних стилів, а також впроваджувати в текст малюнки і вікна;

Фреймворк Tkinter є стандартною утилітою мови програмування Python і це спрощує час її налаштування, проте бібліотека PyQt є складовою одного з найпопулярніших графічних фреймворків Qt і зважаючи на те що цей фреймворк написаний мовою програмування C++, швидкість виконання поставлених задач в PyQt буде вищою. Тому саме ця бібліотека була обрана для розробки керуючої системи засобів автоматизації складських виробничих процесів.

Також слід зазначити, що PyQt має значно більше функціональних можливостей, що дозволить розширювати систему при подальшій розробці. Як приклад, в PyQt є віджет для роботи з таблицями, який знадобиться при створенні звітів про користування засобами автоматизації складських виробничих процесів, а також для створення нових збірок, де також необхідний табличний вигляд при створенні інтерфейсу.

Бібліотека OpenCV

Оскільки серед поставлених задач є задача розробки програми з можливістю здійснення фото-запису елементів для створення збірки, якою користуватиметься працівник складу, потрібно обрати релевантний інструмент для під’єднання програмних засобів із камерою, встановленою біля робочої станції. Найбільш розповсюдженою та широковживаною є бібліотека OpenCV.

OpenCV (англ. Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим кодом) — бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає

засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Бібліотека розроблена Intel і нині підтримується Willow Garage та Itseez. Сирцевий код бібліотеки написаний мовою C++ і поширюється під ліцензією BSD. Біндинги підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua та інших. Може вільно використовуватися в академічних та комерційних цілях.

Бібліотека містить понад 2500 оптимізованих алгоритмів, серед яких повний набір як класичних так і практичних алгоритмів машинного навчання і комп'ютерного зору (рис 2.5).

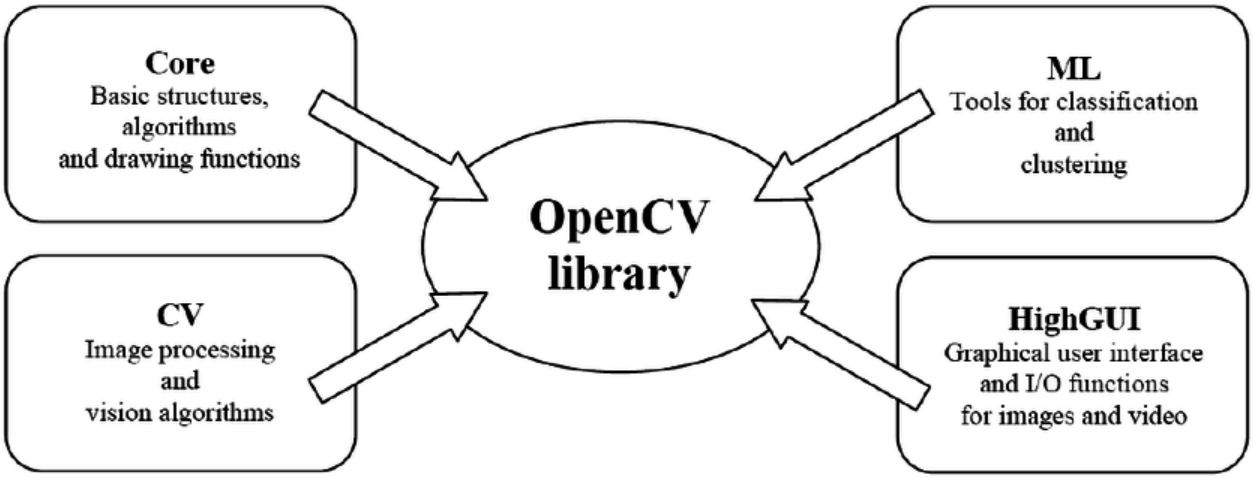

Рисунок 2.5 – Розділи бібліотеки OpenCV

Алгоритми OpenCV застосовують у таких сферах:

- Аналіз та обробка зображень;
- Системи з розпізнавання обличчя;
- Ідентифікації об'єктів;

- Розпізнавання жестів на відео;
- Відстежування переміщення камери;
- Побудова 3D моделей об'єктів;
- Створення 3D хмар точок зі стерео камер;
- Склеювання зображень між собою, для створення зображень всієї сцени з високою роздільною здатністю;
- Система взаємодії людини з комп'ютером;
- Пошуку схожих зображень із бази даних;
- Усування ефекту червоних очей при фотозйомці зі спалахом;
- Стеження за рухом очей;
- Аналіз руху;
- Ідентифікація об'єктів;
- Сегментація зображення;
- Трекінґ відео;
- Розпізнавання елементів сцени і додавання маркерів для створення доповненої реальності та інші.

Засоби комунікації з виконавчою системою

PySerial – це модуль для мови Python, який надає підтримку послідовним з'єднанням (RS-232) на різних пристроях: послідовні порти старого типу, Bluetooth-модулі, інфрачервоні порти тощо. Він також підтримує віддалені послідовні порти через RFC 2217.

Цей модуль працює з програмами написаними мовою Python, що працюють на Windows, OSX, Linux, BSD (будь-яка POSIX-сумісна система) і IronPython. Він випущений за ліцензією на безкоштовне програмне забезпечення.

Особливості PySerial:

- Інтерфейс одного й того ж класу на всіх платформах, що підтримуються.
- Доступ до налаштувань порту за допомогою властивостей Python.
- Підтримка різних розмірів байтів, стоп-бітів, паритету та регулювання потоку за допомогою RTS / CTS та / або Xon / Xoff.
- Робота з тимчасовим прийомом або без нього.
- Також підтримується файл типу API з "read" та "write" ("readline" тощо).
- Файли в цьому пакеті написані повністю мовою Python.
- Порт встановлений для бінарної передачі. Ніяких байтів NULL, CR-LF перекладу тощо (що використовуються в POSIX.) Це робить цей модуль універсально корисним.

Система управління базами даних

Для зберігання інформації про збірку та місцезнаходження фотографій елементів збірки об'єктивно необхідно використовувати зовнішню систему управління базами даних.

MySQL – вільна реляційна система управління базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за замовленням ліцензійних користувачів.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP, VertrigoServ. Звичайно MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Протокол передачі

File Transfer Protocol (FTP) – це протокол, який використовується для передачі файлів між FTP-сервером та FTP-клієнтом, роль якого найчастіше виконує комп'ютер, підключений до мережі (рис. 2.2). FTP найчастіше використовується для завантаження файлів в мережу. Це альтернативний вибір HTTP-протоколу для вивантаження та завантаження файлів на FTP-сервери.

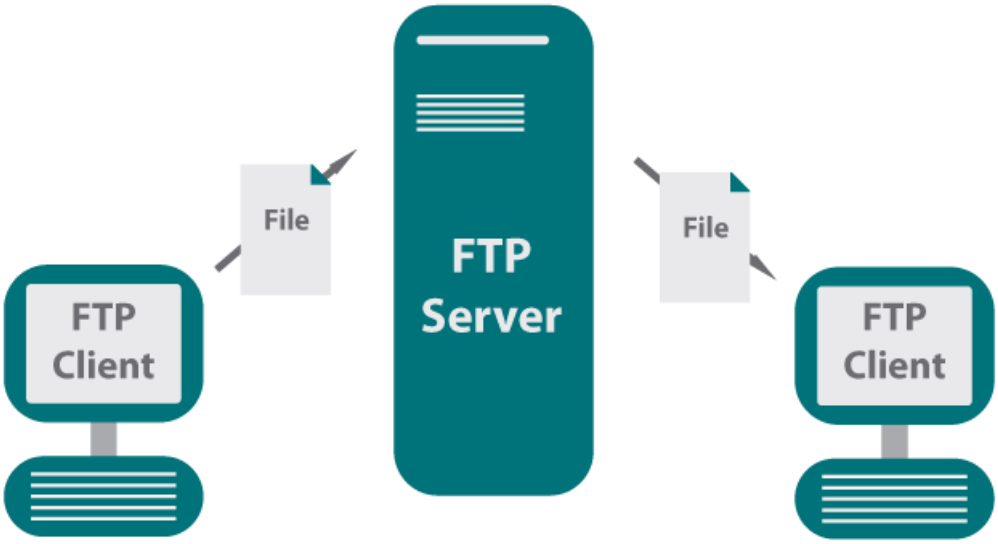

Рисунок 2.2 – Схема роботи FTP протоколу

Протокол FTP, що використовується для стандарту Інтернету, був складені Комітетом цільової групи в галузі Інтернету як серія формальних документів RFC (Request for Comments). На сьогодні цей документ був відредагований декілька разів для вдосконалення протоколу FTP.

FTP був створений з загальною метою дозволити непряме використання комп'ютерів у мережі, полегшивши для користувачів для переміщення файлів з одного місця в інше. Як і більшість протоколів TCP / IP, він заснований на моделі клієнта / сервера, при цьому FTP-клієнт на користувальницькій машині створює з'єднання з FTP-сервером для надсилання та отримання файлів на сервер і з нього. Основними завданнями FTP було спростити передачу файлів, а також захистити користувача від деталей реалізації того, як фактично переміщуються файли з одного місця в інше. З цією метою FTP призначений для автоматичного вирішення багатьох проблем, які потенційно можуть виникнути через різницю форматів файлів, що зберігаються в різних системах.

Після встановлення TCP-з'єднання створюється керуючий зв'язок FTP. Внутрішні команди FTP передаються

за цим логічним з'єднанням на основі правил форматування, встановлених протоколом Telnet. Кожна команда, відправлена клієнтом, отримує відповідь від сервера, щоб вказати, чи це було успішним або не вдалося. Підключення даних встановлюється для кожної передачі даних, яку потрібно здійснити. FTP підтримує або звичайне, або пасивне з'єднання даних, що дозволяє сервер або клієнт ініціювати з'єднання даних. Кілька типів даних і типів файлів підтримуються, щоб забезпечити гнучкість для різних типів передач.

Щоб забезпечити передачу та отримання файлів без втрати даних, які можуть пошкодити їх, FTP використовує надійний протокол управління передачею (TCP) на транспортному рівні. Система автентифікації використовується для забезпечення доступу лише до авторизованих клієнтів до сервера. У той же час функція, яка іноді називається анонімним FTP, дозволяє організації, яка бажає, створити загальний інформаційний сервер для надання файлів будь-кому, хто хоче їх завантажити. Інтерфейс між користувачем FTP і протоколом забезпечується у формі набору інтерактивних команд користувача.

Після встановлення зв'язку та завершення аутентифікації для надсилання або отримання файлів може бути використано дві основні команди. Додаткові команди підтримки надаються для керування FTP-з'єднанням, а також для виконання функцій підтримки, таких як перелік вмісту каталогу або видалення або перейменування файлів. Останнім часом створено графічні реалізації FTP, що дозволяє користувачам передавати файли за допомогою клацань миші замість запам'ятовування команд. FTP також може бути використаний безпосередньо іншими програмами для переміщення файлів з одного місця в інше.

FTP використовується для:

• Завантаження веб-сторінок на веб-сервери для публікації в мережі;

• Перегляд та завантаження файлів із загальнодоступних програмних сайтів;

• Передача великих файлів між двома сторонами, що є занадто великими для вкладень електронної пошти;

Для використання FTP необхідне FTP-клієнтське програмне забезпечення та FTP-сервер. Також потрібно знати адресу сервера, ім'я користувача та пароль і номер порту.

З’єднання з FTP-протоколом відбувається за допомогою двох TCP портів для всіх комунікацій між сервером і користувачем:

• COMMAND Port: це основний TCP-порт, створений під час сеансу з’єднання. Він використовується для передачі команд і відповідей. Порт 21 (незабезпечений) або 990 (захищений) - стандартні командні порти, які використовуються.

• DATA Port: кожен раз, коли файли або каталоги переносяться між сервером і клієнтом, встановлюється випадкове TCP-з'єднання даних і передача даних починається через з'єднання. Після завершення передачі даних з'єднання буде закрито. Подальші з'єднання даних встановлюються та припиняються за необхідністю. З'єднання даних ніколи не залишаються відкритими.

FTP передає файли між системами за допомогою одного з двох режимів - ASCII або двійкового. Режим визначається на початковій стадії всіх транзакцій FTP на сервері. FTP-клієнт автоматично переключається в необхідний режим.

Режим ASCII використовується виключно для передачі тексту та HTML файлів. Двійковий режим передає zip-файли, зображення або виконувані файли в двійковій формі. Двійкові файли не можуть бути

відправлені через режим ASCII, і навпаки, оскільки відбувається пошкодження даних.

Висновки до розділу 2

Проаналізовано технології та засоби розробки, що можуть бути використані при побудові керуючої системи засобів автоматизації складських виробничих процесів. Як результат аналізу вирішено:

- Розробляти керуючу систему на базі мінікомп'ютера Orange Pi PC, щоб задовільнити вимогам щодо зниження енергозатратності керуючої системи та її собівартості.
- Використати клієнт-серверну архітектуру, оскільки такий шаблон є домінуючою концепцією у створенні розподілених мережних додатків.
- Використати MVC як паттерн програмування. Його використання суттєво додає гнучкості при дизайні програмного забезпечення та полегшує подальші зміни чи розширення програми.
- Для розробки керуючої системи використати мову програмвання Python, що зумовлено наявністю зручних інструментів для здійснення комунікації з виконавчою системою, а також можливістю крос-платформної розробки при подальшому розвитку системи.
- У якості засобів розробки графічного інтерфейсу користувача використати фреймворк PyQt, що зумовлено наявністю великої кількості функціональних можливостей.
- Використати бібліотеку комп'ютерного бачення OpenCV, оскільки однією з вимог до керуючої системи є створення фотографій елементів збірки шляхом використання камери.
- Використати MySQL Server як систему управління базами даних. Наявність значної кількості типів таблиць дозволить

здійснювати додаткові операції. База даних буде містити інформацію про збірки, елементи та місце знаходження фотографій.

- Використати протокол FTP для передавання та зберігання створених відображень елементів.
- Використати модуль PySerial для комунікації з виконавчою системою за допомогою віртуальних портів.

3. ОПИС РОЗРОБЛЕНИХ ЗАСОБІВ

3.1 Налаштування мінікомп’ютера

Перш за все для використання керуючої системи потрібно налаштувати технічне обладнання.

Керуюча система розробляється з метою використання її на мінікомп’ютері Orange Pi, оскільки саме із застосуванням такого рішення вирішено будувати засоби автоматизації складських виробничих процесів.

Аналіз функціональних вимог до керуючої частини засобів автоматизації складських виробничих процесів дозволив визначити склад програмного забезпечення, яке потрібно встановити на мінікомп’ютер. У якості операційної системи було обрано Armbian, що базується на ядрі Linux з фреймворком Debian.

Мінікомп’ютер Orange Pi забезпечує керуючу систему усім необхідним, а саме: встановленою операційною системою, процесором з достатньою потужністю для виконання функцій керуючої системи, налаштованими портами для комунікації з периферійними пристроями, як користувацькими (монітор, клавіатура і миша), так і виробничими (сенсорні датчики, світлодіоди та ін.), вихід в мережу та підключення до бази даних, можливе підключення камери тощо. Тобто таке використання мінікомп’ютера Orange Pi за своїми функціональними можливостями не поступається значно дорожчому технічному обладнанню.

Визначено наступну послідовність дій, яку необхідно й достатньо виконати, що забезпечити коректну роботу мінікомп’ютера та його подальшу підтримку, розиток та оновлення керуючої системи:

1. Завантажити Armbian. Змонтувати образ на microSD картку.
2. Встановити образ Armbian з microSD картки на постійну пам’ять EMMC мінікомп'ютера.

3. Оскільки керуюча система написана мовою програмування Python – завантажити необхідні пакети цієї мови за допомогою системи керування пакетами PIP.
4. Встановити pySerial – модуль Python, який інкапсулює доступ до послідовного порту. Він надає доступ програмному рівню до портів для різних платформ і реалізацій мови Python.
5. Встановити PyQT – модуль для запуску інтерфейсу користувача керуючої системи.
6. Налаштувати роботу з віртуальним USB-портом для підключення датчиків периферійних пристроїв.
7. Завантажити та встановити LFTP – консольний FTP-клієнт для UNIX і UNIX-подібних операційних систем.
8. Завантажити та налаштувати MySQL Server: сконфігурувати сервер в файлі /*etc*/*mysql*/*mysql*.conf.d/*mysqld.cnf* та надати доступ користувачу до бази даних.

Визначені вище дії повністю налаштовують мінікомп’ютер і операційну систему Armbian на використання програмами керування складськими процесами, забезпечують необхідний функціонал системи та його відповідність технічним вимогам.

3.2 Архітектура програмних засобів системи автоматизації

Розроблені засоби автоматизації складських виробничих процесів використовують загальну архітектуру типу клієнт-сервер. При розробці застосовувався шаблон програмування MVC. Засоби автоматизації було поділено на керуючу та виконавчу системи (рис. 3.1).

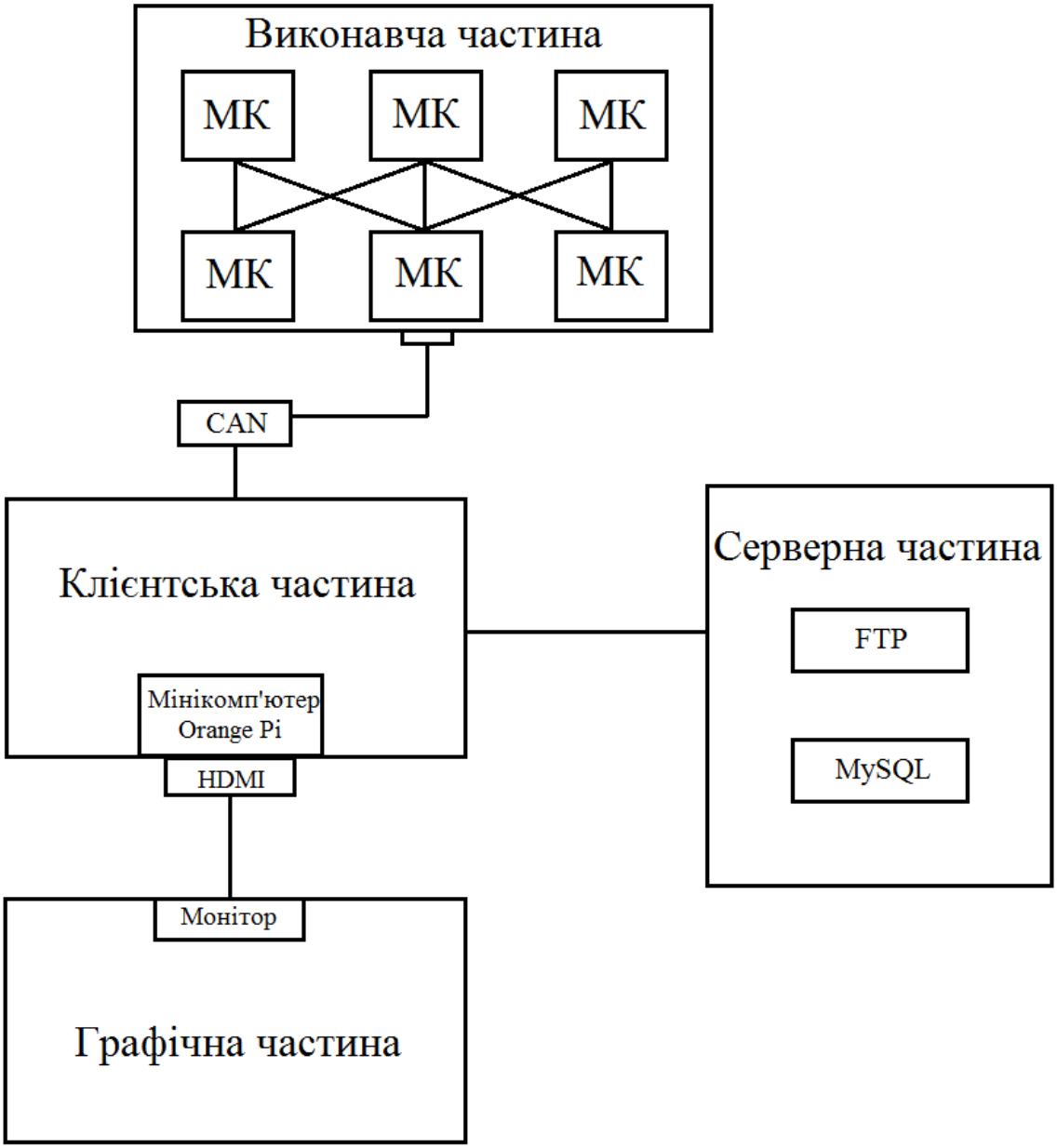

Рисунок 3.1 – Загальна архітектура засобів автоматизації складських виробничих процесів

Виконавча система складається з системи мікроконтролерів, які з'єднуються з керуючою частиною через шину даних, кінцевим виходом якої є віртуальний порт, розглянутий в попередньому розділі та застосований при розробці мінікомп'ютера Orange Pi.

Розроблені засоби включають наступні складові:

1. виконавча частина;

2. клієнтська частина;
3. серверна частина;
4. графічна частина.

Клієнт-серверна архітектура в поєднанні з архітектурним шаблоном MVC (рис. 3.2).

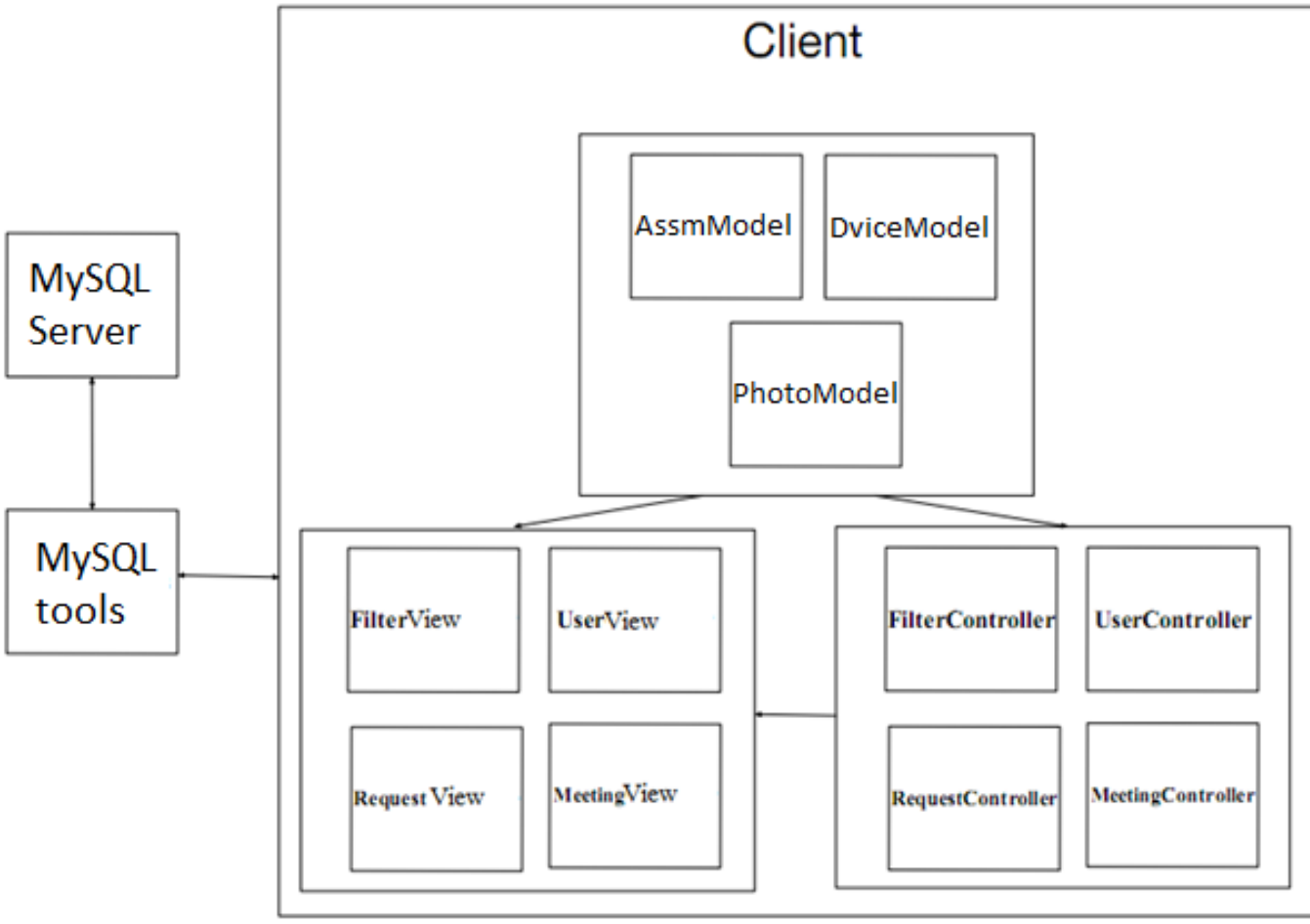

Рисунок 3.2 – Клієнт-серверна архітектура в поєднанні з архітектурним шаблоном MVC

Серверна частина

В магістерській дисертації в якості бази даних використовується MySQL Server. База даних – це сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. База даних містить схеми, таблиці,

подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

База даних MySQL Server на поточному етапі розміщена на веб-ресурсі pythonanywhere.com, де і відбувається керування нею. Для її використання потрібно підключити фреймворк Qt4. Він містить бібліотеку QSQL, яка дозволяє керувати різними базами даних, зокрема MySQL.

Після реєстрації проекту на веб-ресурсі розробнику надається IP-адреса, яку він використовує в проекті. Реєстрація бази даних в проекті представлена наступним кодом:

```
db = QSqlDatabase("QMYSQL")
db.setHostName("host")
db.setDatabaseName("PicktoLight")
db.setUserName("name")
db.setPassword("password")
db.open()
```

Після введення хосту, назви бази та паролю до неї програмний засіб отримує доступ до бази даних і може здійснювати запити, які описані у бібліотеці QSQL.

Таблиці бази даних відображаються в програмному ресурсі MySQL Workbench. MySQL Workbench надає зручний спосіб створення окремої моделі в базі даних, а також її полів.

База даних програмного модулю для створення збірки містить чотири таблиці (рис. 3.3).

Таблиця UserInfo містить такі поля, як: ім’я користувача, статус його активності, а також його ідентифікаційний номер. Після авторизації користувача відбувається отримання його даних для дослідження його активності.

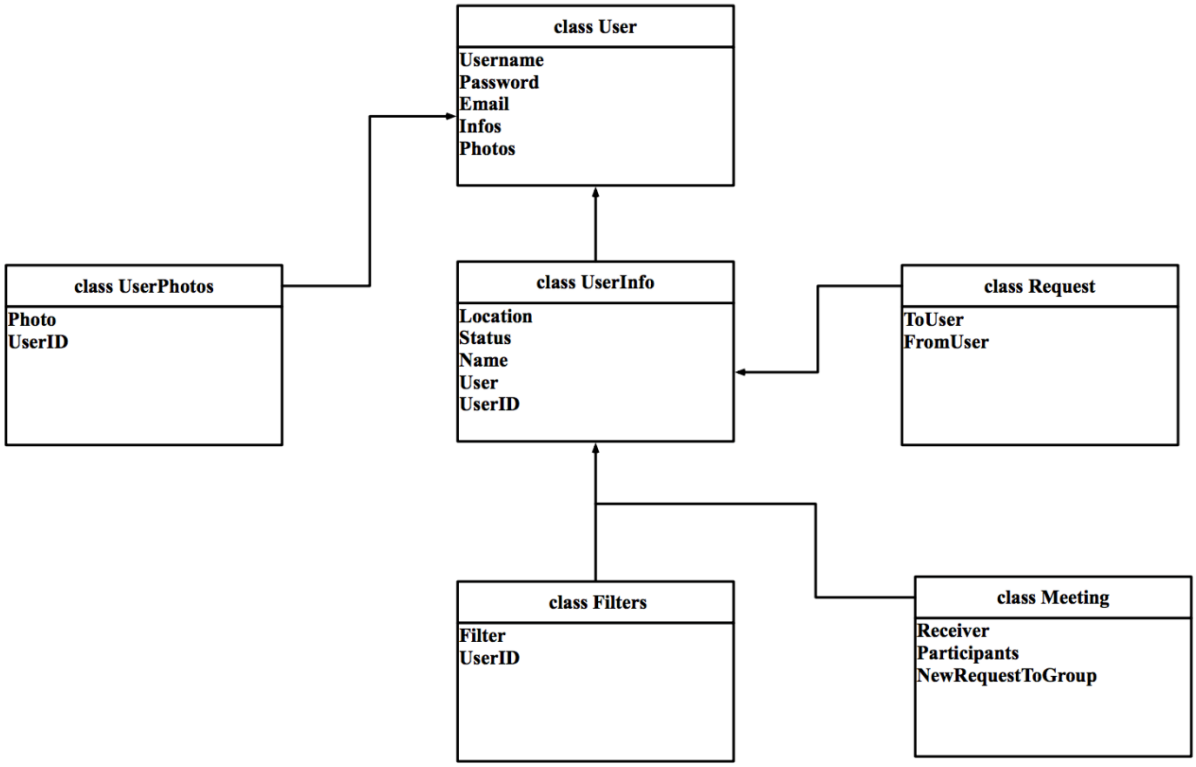

Рисунок 3.3 – База даних програмних засобів складських виробничих процесів

Таблиця AssemblyInfo містить такі поля як:

1) назва збірки;
2) номер кроку;
3) ідентифікатор апаратного пристрою, який відповідає за комірку, в якій розташовується товар чи елемент збірки.
4) поле картинки з відображенням елемента всього пристрою;
5) поле картинки з відображенням результату, тобто ілюстрація того, що працівник повинен виконати з елементом;
6) поле для коментарів, текстових вказівок працівникові на даному етапі;
7) кількість елементів.

Таблиця DeviceInfo зберігає інформацію про назву елемента та його ідентифікаційний номер. Це необхідно для того, щоб користувач міг отримати файл з розташуванням елементів на складі.

Таблиця History – основна таблиця бази даних. Вона містить такі поля як:

1) ім’я працівника, який виконує роботу;
2) назва збірки, яка створюється працівником;
3) час, який витратив працівник на виконання збірки;
4) час перерви працівника.

Ця таблиця містить найважливішу інформацію, яка необхідна для дослідження ефективності працівників в подальшому. За допомогою цих даних можна робити денні чи тижневі звіти, а також будувати математичні моделі для визначення показників ефективності, які найбільше впливають на робочий процес, що передбачається зробити в подальшому.

3.3 Розроблене програмне забезпечення

Клієнтська частина програмного забезпечення для формування збірок

Програмне забезпечення для формування збірок складається з п’ятьох складових (рис. 3.4).

Робота програмного забезпечення починається з модуля запуску, який під час запуску відображає логотип та авторські права програми через графічний інтерфейс, а також підвантажуються основні фреймворки та інші засоби роботи з програмою.

Після модуля запуску в програмі відбувається перехід до основного модуля програми – модуль створення збірки. Основним елементом цього модулю є таблиця, реалізована за допомогою QTableView. Вона виконує функцію створення збірок та відображення і редагування вже існуючих.

Також цей модуль містить елементи для видалення збірки та очистки таблиці.

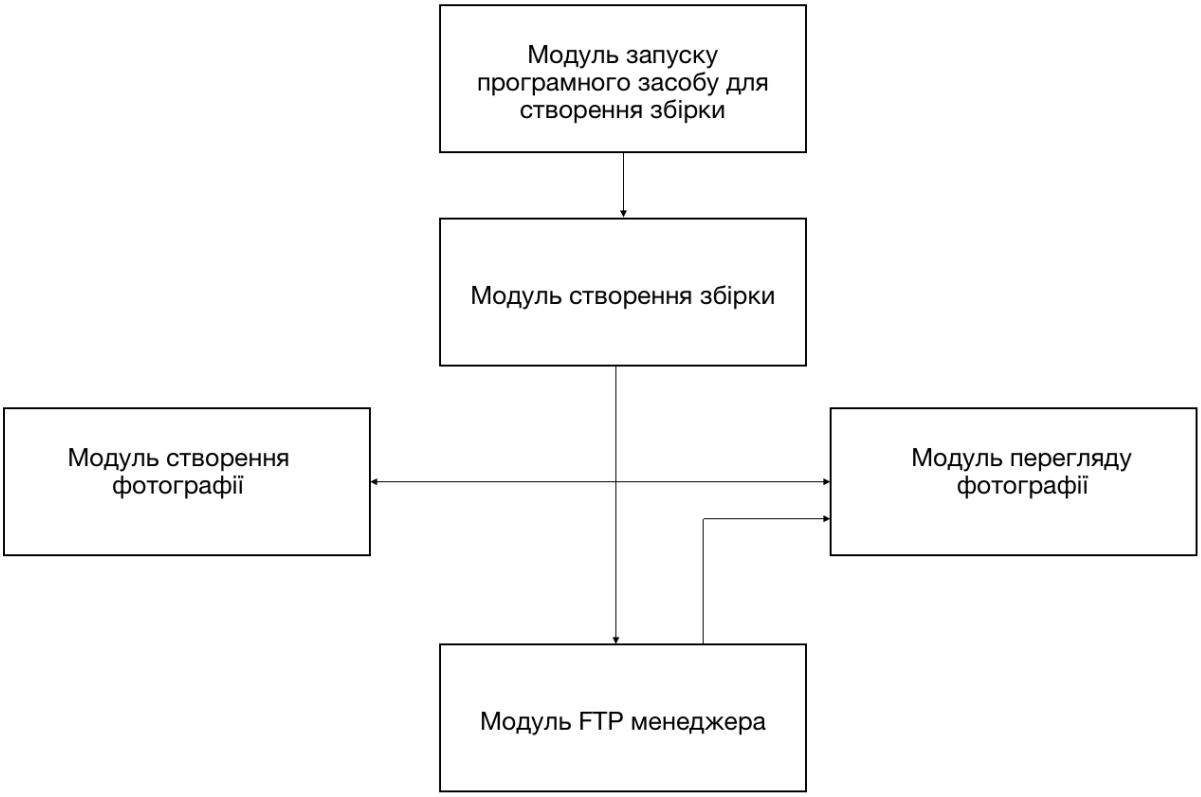

Рисунок 3.4 – Складові програмного забезпечення для формування збірок

Модуль формування збірки робить перехід до трьох інших модулів:

1) Модуль FTP менеджера. Цей модуль виконує зв’язок програмного забезпечення з FTP сервером, який містить існуючі фотографії інших збірок, які можуть бути використані для створення нової збірки.
2) Модуль перегляду фотографії. Цей модуль відображає фотографію яку, користувач захотів переглянути. Цей модуль може викликаним як з модуля створення збірки, так і з модуля FTP менеджера.
3) Модуль створення фотографії. Цей модуль транслює картинку, яку створює фотокамера і за відповідних умов робить фотографію, яка одразу ж відправляється на сервер.

Клієнтська частина програмного забезпечення для виконання збірки

Програмний засіб виконання збірки складається з п’ятьох складових (рис. 3.5). Робота програмного забезпечення для виконання збірки починається з модуля запуску, який під час запуску відображає логотип та авторські права програми за допомогою графічного інтерфейсу, а також підвантажуються основні фреймворки та інші засоби роботи з програмою.

Після модуля запуску додатку відбувається перехід до основного модуля програми – модуль авторизації працівника і вибору збірки для роботи. В цьому модулі користувач обирає своє ім’я та вводить свій пароль для авторизації. Також він обирає збірку, яку хоче виконати і пограмне забезпечення переключається на модуль формування збірки.

Рисунок 3.5 – Складові програмного забезпечення виконання збірки

Модуль створення збірки – найважливіший модуль програмного забезпечення. Він послідовно відображає кожен крок створення збірки,

надаючи працівнику найважливішу інформацію на даному кроці, таку як фотографії та текстові інструкції того, що має зробити працівник на даному кроці.

Модулі створення файлів статистики працівника та статистики виконаних збірок дають можливість створення .csv файлів, які надають інформацію про те скільки збірок виконав працівник за визначений час та скільки конкретних збірок виконали всі працівники відповідно.

3.4 Розроблений інтерфейс

Інтерфейс програмного забезпечення для формування збірки

Керуюча система засобів автоматизації складського виробничого процесу передбачає можливість створення, редагування та видалення збірок нових товарів чи замовлень. Для зручності створення нової збірки це програмне забезпечення повине мати змогу взаємодіяти з виконавчою частиною. Зважаючи на те, що виконавча частина має змогу працювати в декількох режимах, при запуску програмного забезпечення він має попередити виконавчу частину про її перехід в режим навчання. Завдяки цьому при додаванні нового кроку при створенні замовлення (збірки) і вказівці ідентифікатора апаратного пристрою створюється процес, що відправляє команду для сигналізації місцезнаходження даного пристрою.

Основний екран (рис. 3.6) для створення нової збірки містить табличне поле з такими елементами:

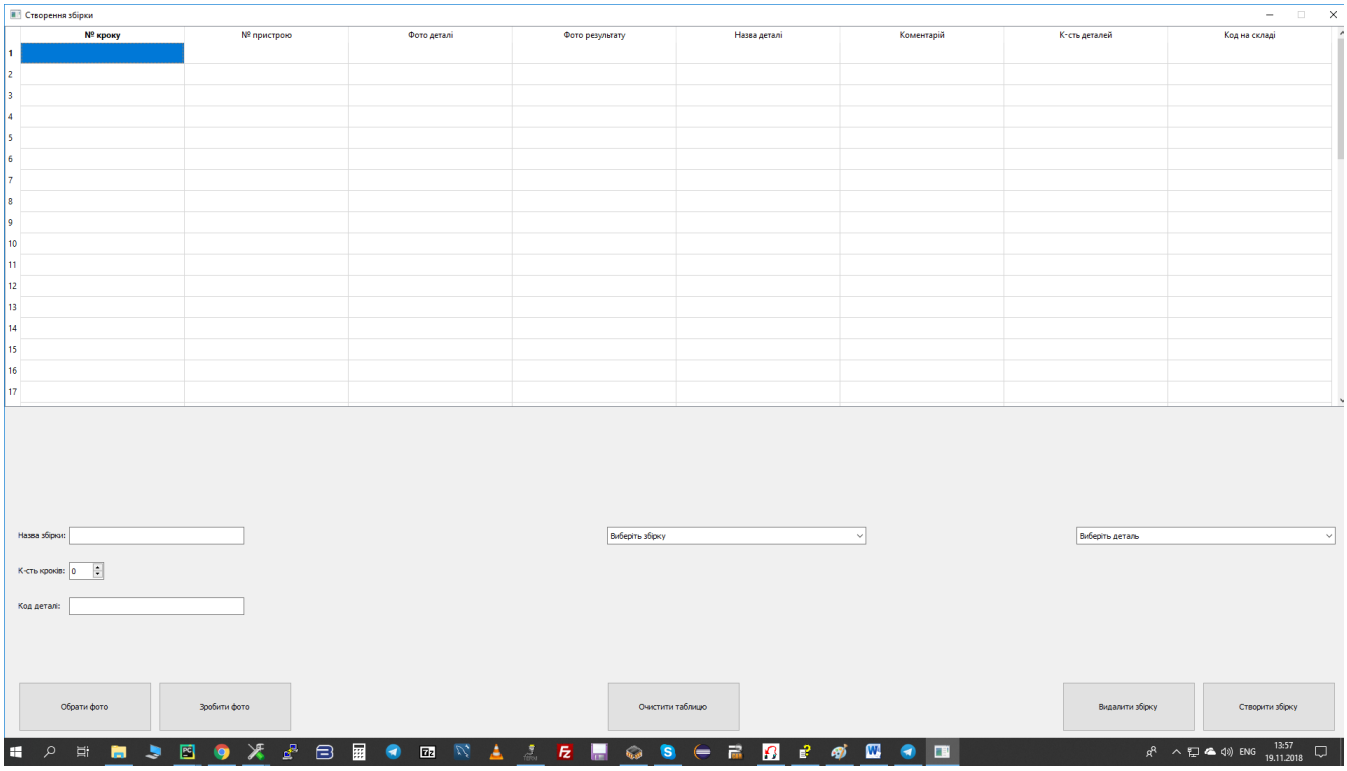

Рисунок 3.6 – Основний екран

1) номер кроку;
2) ідентифікатор апаратного пристрою, який відповідає за комірку, в якій знаходиться елемент;
3) поле картинки елементу всього пристрою;
4) поле картинки результату, тобто ілюстрація того, що працівник повинен виконати з елементом;
5) поле для коментарів, текстових вказівок працівникові на даному етапі;
6) кількість елементів;
7) номер елементу на складі.

Користувач вводить ідентифікатори пристрою в десятковому вигляді, але при формуванні збірки цей ідентифікатор перетворюється в рядок, який відповідає протоколу спілкування з виконавчою частиною. Код функції перетворення значення, введеноко користувачем:

```
def make_connect_device(self, device):
```

```
        """
        :param device: value from '№ присторю'
        :return: '$$hex(device)' if it is less that 255 and is digit or -1
        """
        if device.isdigit():
            if int(device) <= 255:
                if int(device)<16:
                    device = '$$0{0};'.format(hex(int(device))[2:]).upper()
                    return device
                else:
                    device = '$${0};'.format(hex(int(device))[2:]).upper()
                    return device
        return -1
```

Іншими елементами екрану програмного забезпечення створення збірки є:

1) поле введення назви збірки;
2) поле кількості кроків у збірці;
3) випадаючий список існуючих збірок
4) випадаючий список елементів, які використовувались в попередніх збірках;
5) кнопки створення та видалення збірок;
6) кнопки створення нових фотографії, або вибору існуючих;
7) кнопка очистки таблиці.

При формуванні збірки це програмне забезпечення перевіряє правильність введення всіх полів. Ця функція повинна виконуватись саме на етапі створення для уникнення на етапі роботи появи можливих помилок.

Таблиця з елементами збірки має 100 рядків. Для тствоерення збірки, користувач повинен внести в поле кількості кроків число, яке відповідає числу заповнених рядків в таблиці. Такий механізм розроблений для того, щоб працівник не зміг допуститись помилки при заповненні таблиці. Наприклад, якщо при заповнених n кроках в таблиці користувач введе в поле число, яке не дорівнює n, програмний засіб не дозволить йому створити збірку (рис 3.7).

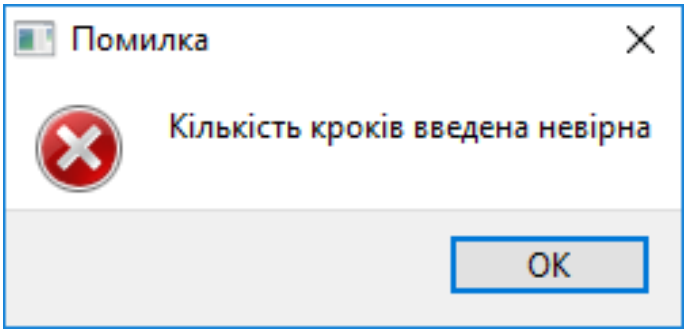

Рисунок 3.7 – Повідомлення при введенні неправильної кількості кроків

Другою помилкою, що може виникнути є незаповнене поле назви збірки (рис 3.8). Це поле бажано заповнювати в першу чергу, оскільки воно також відповідає за створення картинки з відображенням результату.

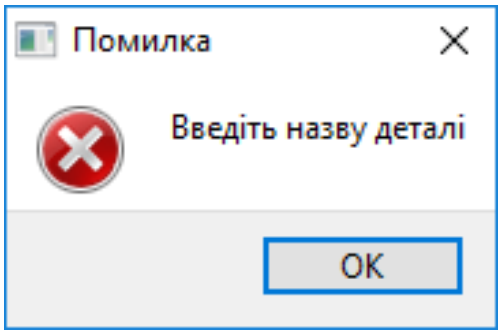

Рисунок 3.8 – Повідомлення при незаповненому полі назви збірки

Користувач має змогу обрати існуючу картинку або створити нову. При ініціалізації програмного забезпечення для створення збірок, кнопка створення картинки (фотографії) деактивована. Щоб створити фотографію, користувач повинен обрати поля картинки з відображенням

елементу, або картинки з відображенням результату. Для відкриття вікна із зображенням з камери при виборі поля «фото деталі», поле «деталь» повинне бути заповнене, оскільки створена картинка буде мати назву деталі. А при виборі поля «фото результату» має бути заповнене поле «назва збірки». Також цей екран не зможе відкритись, якщо не буде підключена камера (рис 3.9).

При виконанні всіх вищеописаних вимог відкривається екран з поточним зображенням камери.

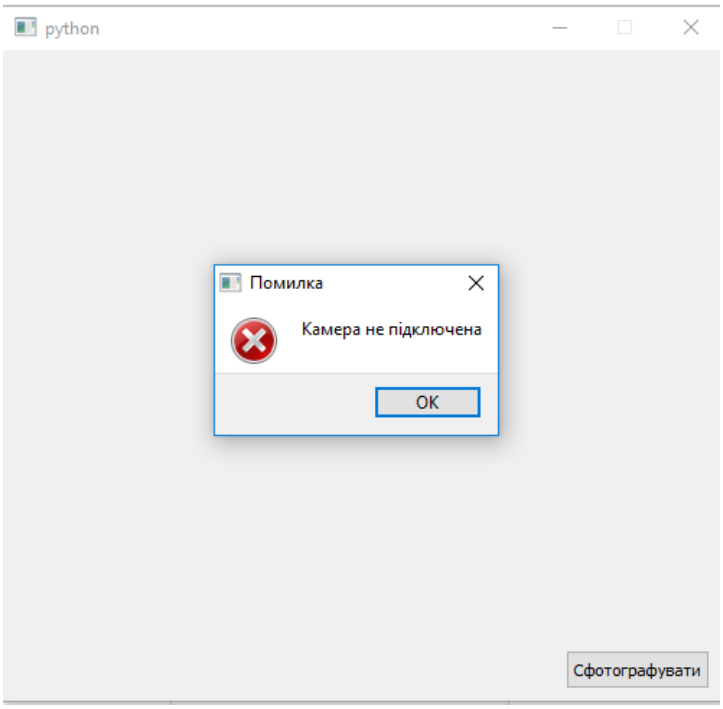

Рисунок 3.9 – Повідомлення при відключеній камері

Після того, як користувач натискає клавішу «зробити фото», створена фотографія відправляється на FTP сервер в папку, яка називається відповідно назві збірки. За роботу зі створенням та обробкою фотографій відповідає бібліотека OpenCV. Процес отримання фреймів, тобто матричних масивів фотографії, є фоновим. Для того, щоб транслювання зображення на екрані не зависало використовується черга, яка кешує дані, поки вони не будуть використані. Функція, що ініціалізує

чергу, заносить в неї фрейми, отримані з камери та транслює зображення з камери на екран, описана нижче:

```
q = queue.Queue()
self.capture_thread = threading.Thread(target=self.grab, args=(0, self.q, 600, 500, 30), daemon=True)
def grab(self, cam, queue, width, height, fps):
    self.capture = cv2.VideoCapture("rtsp://admin:1234abcd@192.168.1.10:554/Streaming/Channels/1")
    self.capture.set(cv2.CAP_PROP_FRAME_WIDTH, width)
    self.capture.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
    self.capture.set(cv2.CAP_PROP_FPS, fps)
    while (self.running):
      self.quit_flag = 1
      frame = {}
      self.capture.grab()
      retval, img = self.capture.retrieve(0)
      frame["img"] = img
      if queue.qsize() < 10:
          queue.put(frame)
      else:
        queue.qsize()
```

При натисканні кнопки «Обрати картинку» запускається екран FTP менеджера (рис. 3.10).

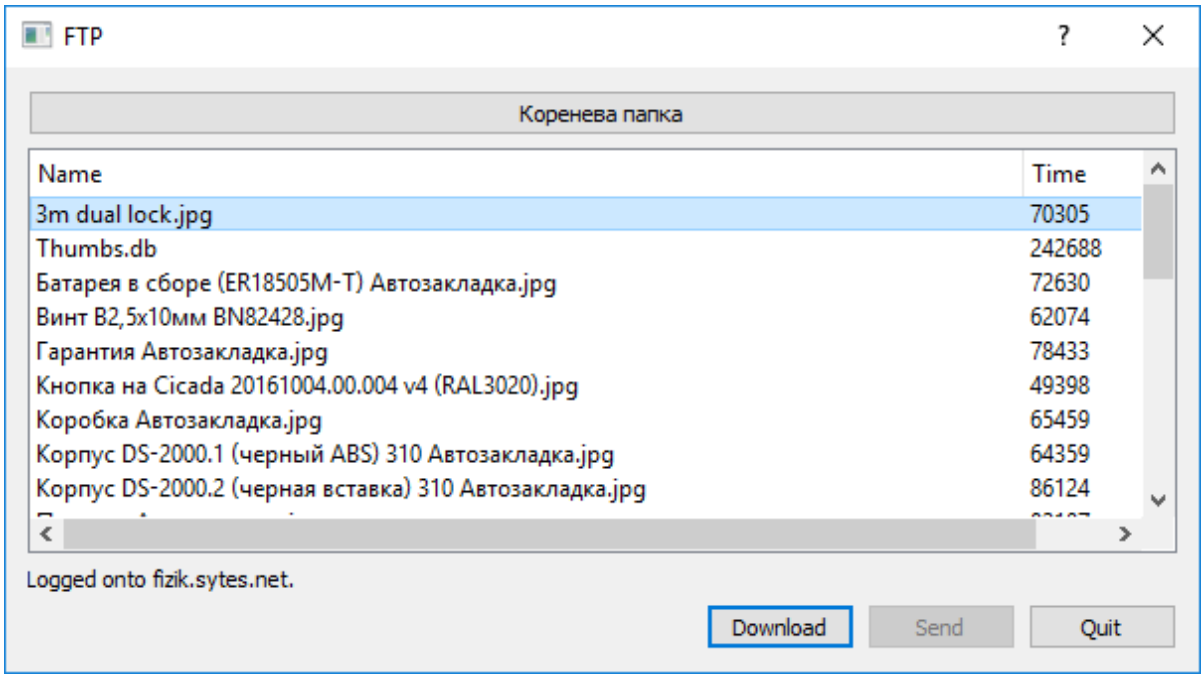

Рисунок 3.11 – Екран FTP-менеджера

Вікно FTP менеджера відображає папки з раніше створеними збірками та всі картинки, які вони містять. При натисканні на картинку відкривається вікно перегляду картинки. При натисканні кнопки «Завантажити» картинка завантажується в змінну об’єкту, що містить папку, в яку необхідно скопіювати картинку та нову її назву, а при натисканні кнопки «Перемістити» переміщує її:

```
def downloadFile(self):
        self.Path = self.fileList.currentItem().text(0)
        self.outFile = QtCore.QFile(self.readyFileName)
        if not self.outFile.open(QtCore.QIODevice.ReadWrite):
            self.outFile = None
            return
        self.ftp.get(self.fileList.currentItem().text(0).encode().decode("latin"),
self.outFile)
        self.progressDialog.setLabelText("Downloading %s..." % fileName)
        self.downloadButton.setEnabled(False)
        self.progressDialog.exec_()
```

```
            self.finalPath = r"{0}\{1}".format(self.assm_name, fileName)
            if self.table.horizontalHeaderItem(self.table.current_col).text() == "Фото деталі":
                self.table.setItem(self.table.current_row, DETAIL_PICTURE_COLUMN, QTableWidgetItem(self.finalPath))
                self.table.setItem(self.table.current_row, DETAIL_COLUMN, QTableWidgetItem(fileName))
            elif self.table.horizontalHeaderItem(self.table.current_col).text() == "Фото результату":
                self.table.setItem(self.table.current_row, ASSEMBLY_PICTURE_COLUMN, QTableWidgetItem(self.finalPath))
            self.putDir = r"//PickToLight/{0}".format(self.assm_name.encode().decode("latin"))
            self.ftp.mkdir(self.putDir)
```

В наведеному вище коді відбувається завантаження картинки на FTP сервер. В якості місця розташування завантаженої картинки обрана назва комплектації замовлення. Якщо на сервері немає такої комплектації, створиться нова папка для збереження картинок комплектацій. Якщо користувач обирає вже існуючу картинку, вона копіюється з папки комплектації, в якій вона знаходиться в нову. При цьому її назва змінюється відповідно до назви елементу або результату (збірки елементів).

Якщо створюється нова картинка, вона з відповідною назвою буде завнтажена на сервер.

Інтерфейс програмного забезпечення для виконання збірки

Програмне забезпечення для роботи зі збірками є основною частиною програмного забезпечення керуючої системи засобів автоматизації складських виробничих процесів. Основною задачею цього

програмного забезпеченняє візуалізація робочого процесу для працівників складу.

Основною вимогою до програмного забезпечення для виконання збірки є його простота у використанні, оскільки воно має бути розрахованим на користувачів без значного попереднього досвіду роботи з комп’ютером.

Початковим вікном додатку є вікно авторизації користувача і вибору збірки, яка виконується (рис 3.11).

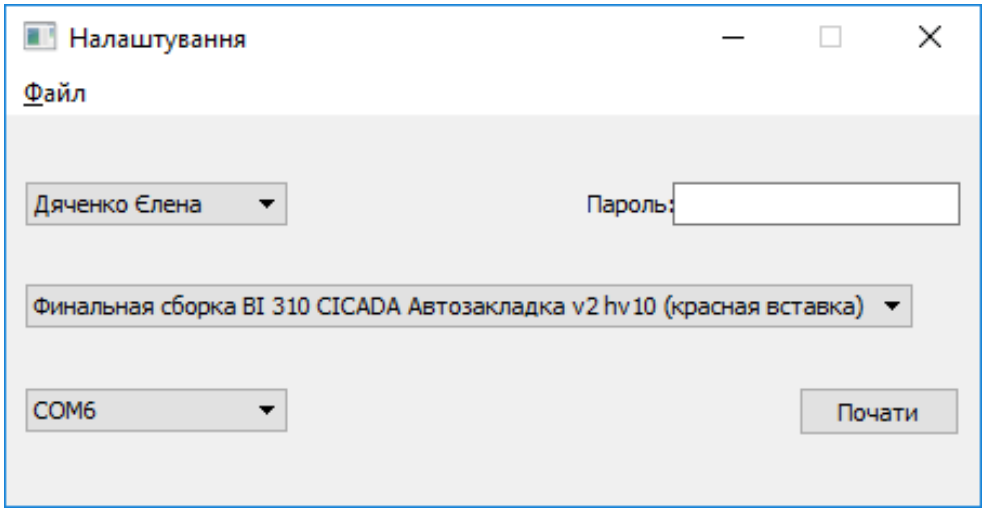

Рисунок 3.11 – Вікно авторизації та вибору збірки

Після введенням користувачем паролю для доступу до в систему та вибору збірки, яку він буде збирати, користувач натискає кнопку “Почати”. Після цього запускається окремий процес, який з’єднується з шиною даних через віртуальний USB COM-порт. Віртуальним USB COM-портом називається програмний інтерфейс, який дозволяє програмам отримувати доступ до пристрою USB, так наче це вбудований послідовний порт. Багато віртуальнич USB COM-портових пристроїв функціонують як “мости”, що роблять перетворення даних між інтерфейсами USB та RS-232 або іншими асинхронними послідовними інтерфейсами.

Операційна система Armbian для Orange Pi PC Plus містить лише один віртуальний USB COM-порт “tty/USB0”. У такому випадку це стає

перевагою, оскільки користувачу не потрібно обирати COM-порт самому – програмний засіб зробить це за нього:

```
from serial.tools import list_ports
COM_ports = [port.device for port in list_ports.comports()]
```

Після успішного під’єднання до шини даних за допомогою бібліотеки PySerial, процес починає опитування присутніх пристроїв на шині. Для цього на шину даних відправляється команда “FE; getpassport;”. Якщо пристрої присутні на шині, вони відреагують на команду повідомленням “FE; 01;”, де 01 – це ідентифікатор пристрою.

Щоб перевірити, чи всі пристрої є на шині, процес отримує дані про збірку, яку обрав користувач, а саме номери пристроїв, які використовуються у збірці. Зважаючи на те, що пристрої виконавчої системи виходять на зв’язок кожні п’ять секунд, спочатку відбувається зупинка процесу перевірки на більший час. Після цього на шину надсилається бінарний рядок b'$$FE; TPASS: 11111; getpassport; \r\n', де «$$FE;» є індикатором звернення до всіх пристроїв, присутніх на шині. Команда «TPASS: 11111;» відповідає за авторизацію керуючої системи у пристрої виконавчої. Команда «getpassport;» потрібна для запиту в пристрою його ідентифікатора.

Після відправлення цього рядку система очікує відповіді. Якщо їй надходять дані, вона знаходить в них ідентифікатори пристуніх пристроїв і перевіряє чи вони відповідають тим пристроям, які вкористовуються в комплектації замовлення. Якщо якийсь пристрій, що використовується в робочому процесі, не вийшов на зв’язок, виконання замовлення (збірки) не відбуденться.

Код функції перевірки пристрою наведений нижче:

```
def _check_devices(self):
        self.sleep(6)
        self.ser.write(b'$$FE; TPASS: 11111; getpassport; \r\n')
```

```
self.sleep(1) #затримака очікування
a = self.ser.readline()
counter = 0
while a != [b'']:
a = self.ser.readline()
 a = a.split(b';')
for el in self.devices:
    if len(a) > 2:
        if len(a) != 1:
            if el[2:3] != b'0':
                if str(int(el[2:4].decode(), 16)) == a[1].decode():
                    counter += 1
                    break
            else:
                if str(int(el[3:4].decode(), 16)) == a[1].decode():
                    counter += 1
                    break
        else:
            if bytes([el[3]]) == a[1]:
                counter += 1
                break
```

Якщо кількість пристроїв, що присутні на шині даних, не відповідає кількості пристроїв, отриманої з бази даних, або відсутні чи присутні пристрої, яких немає на етапах створення збірки, програмне забезпечення не зможе перемкнутись на екран формування збірки (рис. 3.12).

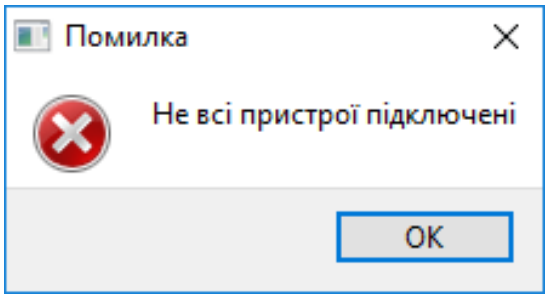

Рисунок 3.12 – Повідомлення при неправильній кількості пристроїв

Зважаючи на те, що при вимкненні апаратного пристрою виконавчої частини, його дані знаходяться протягом наступних п’яти секунд на шині даних, перед запуском першого етапу комплектації відбувається затримка для виконання розпізнавання вимкнутого пристрою (рис. 3.13).

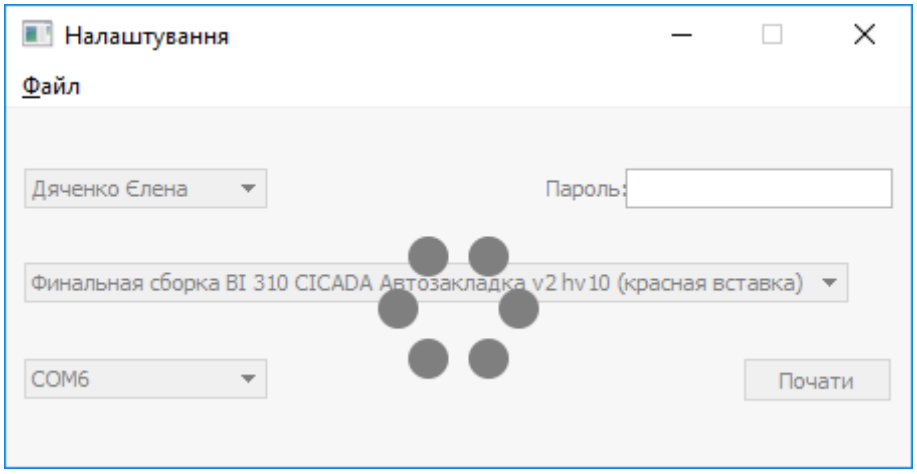

Рисунок 3.13 – Затримка перед запуском

При успішній перевірці активних пристроїв відкривається динамічне вікно відображення поетапних вказівок для створення збірки (рис. 3.14).

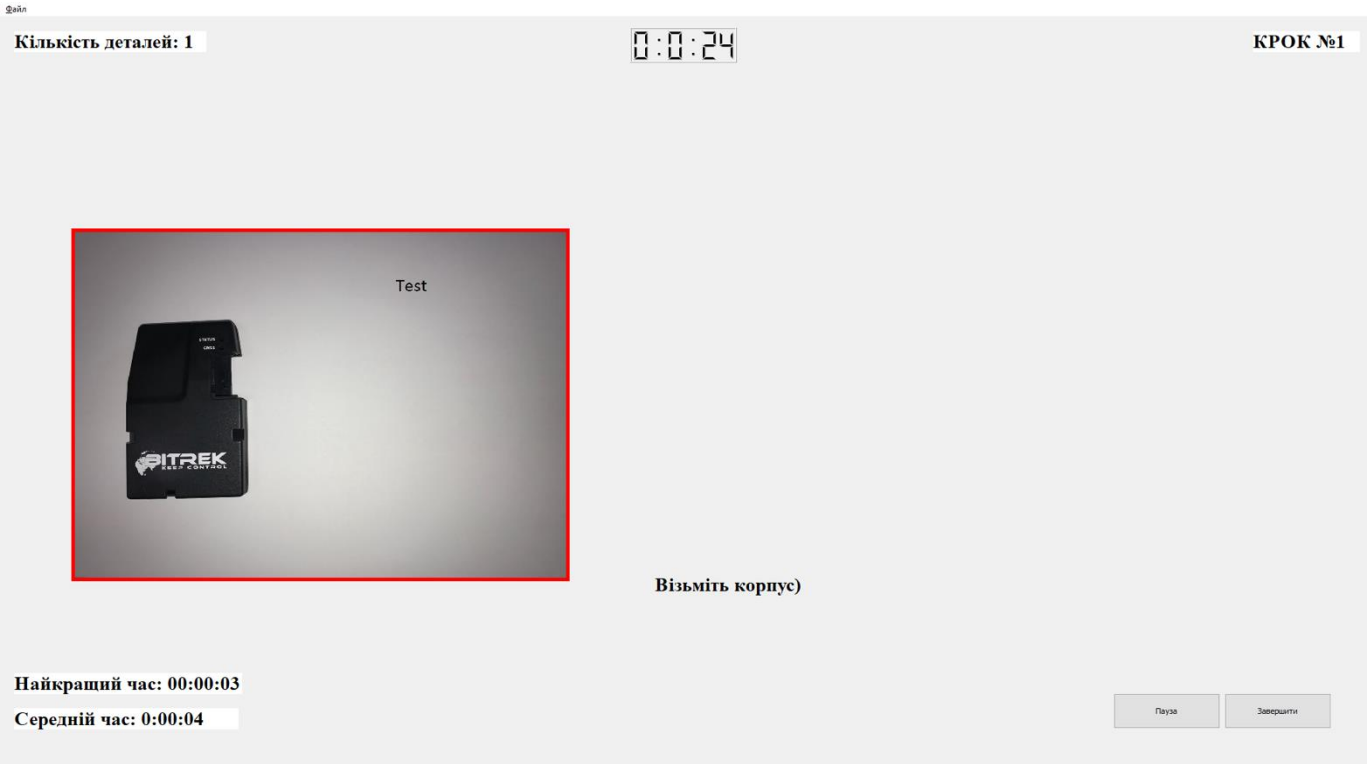

Рисунок 3.14 – Вказівка для створення збірки

При запуску цього вікна, створюється фоновий процес, алгоритм якого виконує декілька важливих функцій:

1. Відображення та зміна фотографій на кожному етапі збірки;
2. Відображення та зміна кількості елементів, коментарів та кроків на кожному етапі збірки;
3. Відображення середнього та найкращого часу виконання збірки та їхнє оновлення після виконання;
4. Відображення часу роботи працівника та зупинка його відліку, якщо працівник поставив роботу на паузу;
5. Неперервна комунікація з виконавчою частиною засобів автоматизації складського виробничого процесу.
6. Занесення відомостей про виконання кожної окремої збірки працівником;

Основною задачею цього процесу є комунікація з виконавчою системою. Комунікація відбувається шляхом відправлення команд, які може обробити виконавча частина. Цими командами є:

1. Light – команда для включення світлового індикатора визначеного апаратного пристрою. Ця команда відправляється разом з ідентифікатором пристрою та символом закінчення команди “;”: “$01; light;”;
2. Dinck – команда для виключення світлового індикатора визначеного апаратного пристрою. Ця команда відправляється разом з ідентифікатором пристрою та символом закінчення команди “;”: “$01; dinck;”;
3. Nextdev – команда, що відправляється активному на поточному етапі пристрою після виконання цього етапу. Після отримання цієї команди апаратний пристрій не буде реагувати на рухи біля його сенсору, оскільки робітник може при необхідності використовувати комірку декілька разів, наприклад, взяти декілька деталей для успішного завершення поточного етапу. Ця команда відправляється разом з ідентифікатором пристрою та символом закінчення команди “;”: “$0A; nextdev;”
4. Getparam – команда, для отримання певних параметрів, які формує виконавча частина. Це можуть бути параметри кількості пауз на пристрої тощо. При формуванні команди враховується ідентифікатор пристрою та ідентифікатор параметра, дані з якого повинен отримати процес: “$01; getparam 200;”;
5. Setparam – команда для налаштування режиму роботи виконавчої системи. При формуванні команди враховується ідентифікатор пристрою, ідентифікатор параметра та нове значення параметра: “$01; setparam 200 2;”.

Отримавши інформацію про збірку з бази даних у вигляді відсортованого за кроками масиву об’єктів моделі Assembly, процес починає обробку даних об’єкта, таких як відображення деталі та результату, коментарів до цього етапу тощо і здійснює їх візуалізацію на екрані. Код отримання даних з бази та формування з них об’єкта моделі Assembly представлений нижче:

```
def get_assembly(self, name):
    req = "select step, dev_name, pic1, pic2, det_count, text, det_name from assembly_process where assm_name = '{0}'".format(name)
    tmp_dict = {}
    if self.is_db_open():
        query = QSqlQuery(self.db)
        if query.exec_(req):
            while query.next():
                step = str(query.value(0))
                tmp = {
                    str(step): {
                        "device": query.value(1),
                        "pictures": [query.value(2), query.value(3)],
                        "details_count": str(query.value(4)),
                        "text": query.value(5),
                        "detail_name": query.value(6)
                    }
                }
                tmp_dict.update(tmp)
            dict = {name: tmp_dict}
            return dict
```

Отримавши ідентифікатор апаратного пристрою на визначеному кроці процес відправляє команду light сигналізації працівнику

місцезнаходження комірки з необхідним елементом на визначеному етапі створення збірки. Якщо апаратний пристрій виконав команду і включив світловий індикатор, він відправляє підтвердження. Якщо підтвердження не прийшло, процес протягом певного часу знову відправляє команду. Якщо підтвердження її виконання не приходить, процес закриває вікно і сповіщує про збій визначеного апаратного пристрою. Якщо підтвердження прийшло, процес очікує на повідомлення завершення етапу.

Після закінчення визначеного етапу робочого процесу, працівник торкається сенсора апаратного пристрою. Він відправляє команду завершення етапу «END». Процес отримує її, відправляє поточному пристрою команду nextdev і переходить до наступного етапу, де відправляє команду light наступному пристрою.

Працівник також може поставити виконання збірки на паузу. Це можна зробити двома способами: натиснути на сенсор апаратного пристрою, що працює в режимі паузи, або натиснувши кнопку “Пауза” на екрані монітора. При цьому пристрою, що був активним, відправляється команда dinck і запам’ятовується поточний крок для коректного відновлення роботи після паузи. Час, який провів працівник в паузі також фіксується для занесення його в базу даних як статистичного показника роботи працівника.

Програмне забезпечення дає можливість отримати такі типи звітів у форматі.csv про виконання робочого процесу (рис. 3.15):

1. Звіт статистики конкретного працініка;
2. Звіт статистики конкретної збірки;
3. Звіт загальної статистики виконання усіх видів збірок усіма працівниками.

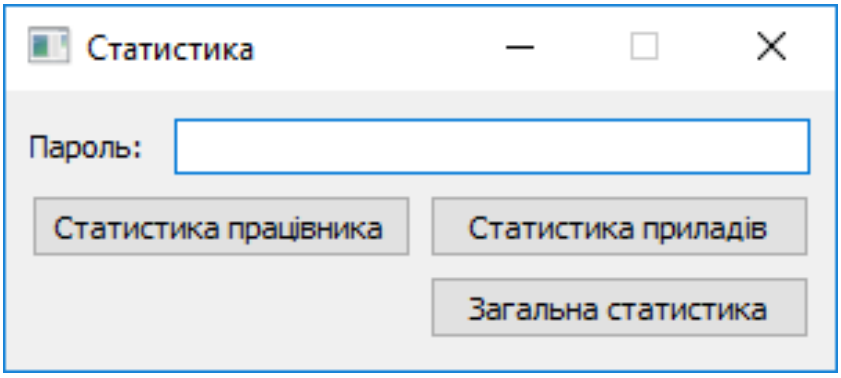

Рисунок 3.15 – Типи звітів

Отримати звіт може працівник з правами адміністратора і відповідним паролем. Якщо пароль був введений три рази неправильно, адміністратору на електронну пошту відправляється лист, про те що з пристрою Orange Pi Pc Plus з визначеною MAC адресою була спроба отримати дані ефективності працівників.

При коректному введенні пароля адміністратор обирає, дані про якого працівника він отримує (рис. 3.16). Після, обирається період, за який потрібно отримати дані (рис. 3.20). В кінці створюється файл формату (*.csv), який надає адміністратору такі дані (рис 3.18):

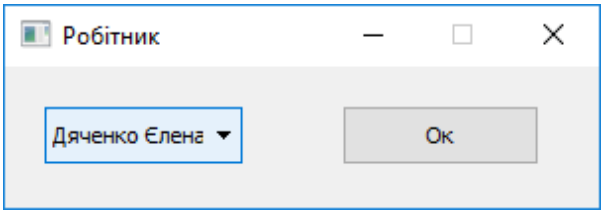

Рисунок 3.16 – Дані про працівника

1. Замовлення, які працівник скомплектував за визначений адміністратором період часу;
2. Кількість замовлень, що були виконані працівником;
3. Загальний час виконання замовлень працівником окремо за кожний робочий день;

4. Середній час виконання замовлень працівником окремо за кожний робочий день;
5. Найкращий час виконання одного замовлення;
6. Кількість зупинок роботи, які здійснив працівник протягом робочого дня;
7. Загальний час всіх зупинок;

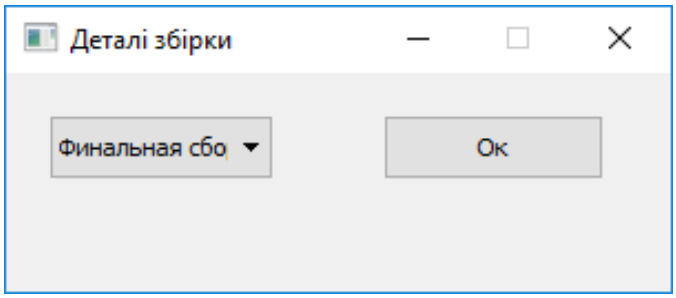

Рисунок 3.17 – Дані про збірки

Період	з	по					
	21.08.18	29.08.18					
Виконав	Юрченко А.А.						
Дата	Збірка	Кількість	Час	Найкращий час	Середній час	Кількість зупинок	Час зупинок
21.08.18	Test1	23	1:45:24	4:19		5	13:21
21.08.18	Test2	23	1:45:24	4:19		5	13:21
21.08.18	Test3	23	1:45:24	4:19		5	13:21
22							
22							
24							
25							
27							
28							

Рисунок 3.18 – Файл з інформацією про виконану роботу конкретним працівником

Якщо адміністратор обирає дані про окрему збірку (рис. 3.17), він отримає схожий з попереднім звіт (рис 3.19).

Відмінністю між ними є те, що другий звіт містить детальну інформацію про окреме замовлення, виконане різними працівниками, в

той час як перший надає детальну інформацію про роботу працівника з різними збірками.

Період	з	по					
	21.08.18	29.08.18					
Збірка	Test1						
Дата	Виконав	Кількість	Час	Найкращий час	Середній час	Кількість зупинок	Час зупинок
21.08.18	Юрченко	23	1:45:24	4:19		5	13:21
21.08.18	Бєсєдін	23	1:45:24	4:19		5	13:21
21.08.18	Мілевський	23	1:45:24	4:19		5	13:21
22							
22							
24							
25							
27							
28							

Рисунок 3.19 – Файл з інформацією про виконану роботу по конкретній збірці

Висновки до розділу 3

При розробленні засобів автоматизації складських виробничих процесів, їх реалізацію було вирішено розділити на керуючу та виконавчу системи. Оскільки керуюча система виконує багато, як основних, так і допоміжних функцій, що задовольняють основним вимогам засобів автоматизації, було розроблене програмне забезпечення для виконання цих функцій.

При розробці керуючої системи запропоновано декілька програмних та апаратних рішень. Керуюча система реалізована на міні комп’ютері Orange Pi PC, що дозволяє зменшити собівартість засобів автоматизації в цілому.

Також розроблене програмне забезпечення, що реалізує такі основні можливості керуючої системи, як створення замовлень, їх реалізацію та моніторинг робочого процесу.

Для комунікації виконавчої та керуючої систем розроблена система команд, завдяки якій керуюча система може налаштовувати виконавчу на

відповідні режими роботи, а також отримувати необхідні для своєї роботи дані від виконавчої системи.

4. РЕЗУЛЬТАТИ ЕКСПЕРИМЕТАЛЬНИХ ДОСЛІДЖЕНЬ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДСЬКИХ ВИРОБНИЧИХ ПРОЦЕСІВ

Експериментальні дослідження проводились із застосуванням тестової версії розробленої керуючої системи в поєднанні із виконавчою, оскільки для проведення повноцінних досліджень потрібні об’єднані обидві системи.

В основу дослідження лягло порівняння часу, необіхдного для здійснення відбору способом використання паперової документації та способом роботи за принципом Pick to light.

Для проведення експеременту двох людей попросили виконати збірку двох комплектів.

Для першої збірки учасникам експерименту було надано паперову документацію, в якій вони могли знайти інформацію про всі елементи збірки та шляхи їх комплектації. Учасники власноруч шукали комірки (ящики) із елементами.

Для другого експерименту ці учасники використовували розроблені засоби автоматизації складських виробничих процесів, які керували процесом збірки комплекту й самі вказували учасникам комірки із елементами.

Після проведення дослідження, порівняльний аналіз показав, що для недосвіченого користувача, значно легшим виявився спосіб відбору з використанням розроблених засобів автоматизації складських виробничих процесів.

Деталі дослідження:

- Відбір з паперовою документацією:
    - Учасник 1, час: 10 хвилин 35 екунд;
    - Учасник 2, час: 9 хвилин 8 секунд.
- Відбір із засобами автоматизації:

- Учасник 1, час: 3 хвилини 56 секунд;
- Учасник 2, час: 3 хвилини 23 секунди.

Висновки до розділу 4

Проведено експериментальні дослідження, в результаті яких встановлено, що людині без попереднього досвіду збирання комплекту знадобиться майже в три рази менше часу для збирання із використанням розроблених засобів автоматизації складських виробничих процесів.

Таким чином результати доводять, що використання саме засобів автоматизаії підвищують ефективність виконання збірок.

ВИСНОВОК

На сьогоднішній день для багатьох підприємств забезпечення прискорення та збільшення якості робочого процесу є важливими вимогами при проектуванні плану роботи. Завдяки технологічному прогресу поява засобів автоматизації складських виробничих процесів дозволяє задовольняти ці вимоги.

З огляду на це в магістерській дисертації було розглянуто декілька найвідоміших підходів для автоматизації системи, таких як Pick to light, Pick by voice тощо. Зважаючи на недоліки та переваги кожного з досліджуваних підходів, було вирішено розробити засоби автоматизації складських виробничих процесів, в основі яких лежаиме принцип Pick to light. Завдяки модифікації окремого функціоналу цього підходу, було досягнуто уникнення його деяких недоліків.

При розробці засобів автоматизації, було вирішено поділити їхню структуру на дві основні системи: керуючу та виконавчу. Керуюча система для засобів автоматизації була розроблена та описана в магістерській дисертації.

Керуюча система являє собою програмний засіб, виконаний за принципом клієнт-серверної архітектури. Серверна частина складається з MySQL сервера, необхідного для додаванн, редагування та видалення замовлень та FTP сервера, необхідного для збрігання медіафайлів (зображень), що стосуються окремих замовлень. Клієнтська частина являє собою UI/UX додаток, який відповідає створення замовлень та роботу з ними. Також вона реалізову протокол з’єднання з виконавчою частиною через віртуальних COM-порт за допомогою попередньо визначеної системи команд.

Реалізація програмних засобів керуючої системи виконувалась високорівневою мовою програмування Python на міні комп’ютері Orange PI PC Plus. Портування керуючої системи на міні комп’ютер знизило

собівартість засобів автоматизації складських виробничих процесів вцілому.

Результати тестових випробувань засобів автоматизації, зокрема керуючої системи, доводять їх функціональність, відповідно до поставлених вимог, ефективність та оптимальність витрат на реалізацію.

Література

1. Warehouse Processes // Режим доступу: https://www.logisticsbureau.com/spotlight-on-7-key-warehouse-processes/
2. Lightning pick: Pick to light systems // Режим доступу: https://lightningpick.com/products/pick-to-light/
3. Pick to light Systems: Cost and ROI // Режим доступу: https://info.voodoorobotics.com/blog/pick-to-light-cost-roi/
4. Курилич Р.А. Комп’ютерний моніторинг і автоматизація складських виробничих процесів / Р.А. Курилич, М.С. Кіндзер, І.П. Дробязко // Прикладна математика та комп'ютинг 2018: зб. тез доп. – К.: Просвіта, 2018. – С.28-33.
5. Orange Pi One – конкурент Raspberry Pi // Режим доступу: https://www.computerra.ru/180397/orange-pi-one/
6. Курилич Р.А. Налаштування мінікомп'ютера для використання керуючою частиною засобів автоматизації складських виробничих процесів/ Р.А. Курилич, І.П. Дробязко // Прикладна математика та комп'ютинг 2018: зб. тез доп. – К.: Просвіта, 2018. – С.28-33.
7. TkInter – Python Wiki // Режим доступу: https://wiki.python.org/moin/TkInter
8. Pyqt – Python Wiki // Режим доступу: https://wiki.python.org/moin/PyQt
9. Armbian – Linux for ARM development boards // Режим доступу: https://www.armbian.com/
10. OpenCV (Open Source Computer Vision Library) library // Режим доступу: https://opencv.org/
11. OpenCV – Wikipedia // Режим доступу: https://en.wikipedia.org/wiki/OpenCV

12. Model–view–controller – Wikipedia // Режим доступу: https://en.wikipedia.org/wiki/Model–view–controller
13. Client–server model – Wikipedia // Режим доступу: https://en.wikipedia.org/wiki/Client–server_model
14. PySerial 3.0 documentation // Режим доступу: https://pythonhosted.org/pyserial/
15. MySQL – Wikipedia // Режим доступу: https://en.wikipedia.org/wiki/MySQL
16. File Transfer Protocol – Wikipedia // Режим доступу: https://en.wikipedia.org/wiki/File_Transfer_Protocol